Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

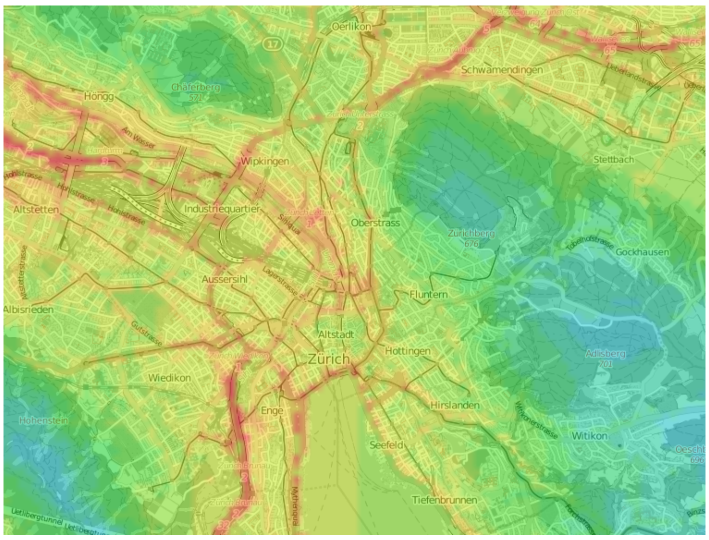
Institut für Technische Informatik und Kommunikationsnetze

DEPARTMENT OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Spring Semester 2017

Visualize Volumes of Air Quality Data

Master Project



Marc Urech
maurech@ethz.ch

25.09.2017

Supervisors: Balz Maag, bmaag@tik.ee.ethz.ch
Dr. Zimu Zhou, zzhou@tik.ee.ethz.ch

Professor: Prof. Dr. Lothar Thiele, thiele@ethz.ch

Abstract

Air quality is an important problem in any populous city, such as Zurich, Switzerland. A few years ago the *OpenSense* project was introduced, which uses low-cost sensors mounted on trams to gather air quality data of high spatial resolution. The data is publicly available over a web-interface which is able to generate simple time-series plots. In its current state the web-tool is not very responsive and presents unfiltered and uncalibrated measurement data. This thesis presents an automated toolchain that cleans up the raw data by filtering and calibrating the sensor measurements in a first step to ensure high quality of the dataset. In a second step, the toolchain precomputes data summaries based on the cleaned up data. Finally, the toolchain delivers visualizations of the air quality data by generating different interactive plots based on cleaned up data and data summaries that are intuitive to understand for the general public and to help people gasp the air quality in Zurich. The visualization part of the toolchain is accessible over a web-interface where it is able to present the data in the form of three different visualizations, namely as trace plots showing measurements on different tram-tracks, as heatmap summaries and as interactive maps showing models of ultrafine particle contamination for the whole city area.

Acknowledgements

I would like to express my gratitude to my supervisors Balz Maag and Dr. Zimu Zhou for the support and continuous inputs and for giving me an insight into their research field. I would also like to thank Prof. Dr. Lothar Thiele for giving me the opportunity to write this masters thesis at the Computer Engineering group at TIK. Further I want to thank Benno Knapp from the traffic control center in Zurich for providing me access to traffic data of the city of Zurich.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 State of the Art

Air quality is an important problem in any populous city, such as Zurich, Switzerland. It is medically proven that atmospheric pollutants have a direct influence on peoples health, can cause several illnesses and even lead to premature death under adverse conditions [2].

Traditionally, air pollution is monitored by a network of static measurement stations operated by official authorities. The extensive cost of acquiring and operating static stations severely limits the number of installations. In the greater Zurich city area, the authorities operate 7 stations[1] which monitor the air quality [1]. The data from these stations can help to answer questions on big and medium scales. E.g. on a big scale, the data helps to compare the air quality to other cities. On a medium scale one can compare the air quality of different districts in Zurich. However, the data can not answer questions on a small scale, e.g. what the influences of certain roads, industrial facilities or demographic parameters are on the air quality inside the city.
In recent years new gas sensors appeared on the market which are inexpensive, small, and suitable for mobile measurements. As part of the *OpenSense* project [9], several such low-cost sensors were integrated into compact air pollution monitoring stations, which are deployed on top of a number of trams in Zurich. The sensors deliver daily megabytes of air quality data, which are stored in a central database. This data is of high spatial resolution as contrasted to the data delivered by static measurement stations. This allows to examine the previously mentioned questions on a smaller spatial scale.

1.2 Motivation

The current functionality of *OpenSense* is visualized in Figure 1.1. As the trams are driving through the city, they collect air quality measurements of different pollutants and store them in a local database. These measurements are periodically transmitted to a central private database. The data is then forwarded to a database accessible for the public over a web-interface, which is able to generate simple time-series plots of raw data.
More details about the sensors, the dataset and the web-interface will be provided in the next sections.

1.2.1 Sensors

Figure 1.2 provides an overview of the sensors deployed on top of trams. The sensors measure the concentration of the pollutants ozone (O_3, Figure 1.2a), carbon monoxide (CO, Figure 1.2b), nitrogen dioxid (NO_2, Figure 1.2c) and ultrafine particulate matter (UFP, Figure 1.2d). The first

[1] **Duebendorf**, **Opfikon**: Balsberg, **Zuerich**: Heubeeribueel, Rosengartenstrasse, Schimmelstrasse, Stampfenbachstrasse, Kaserne

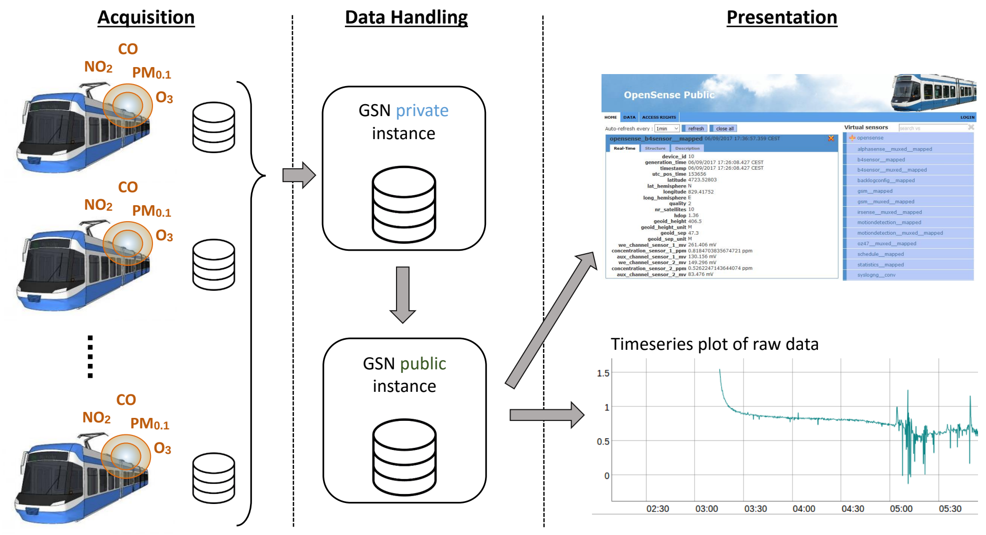
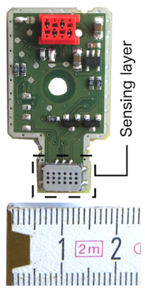
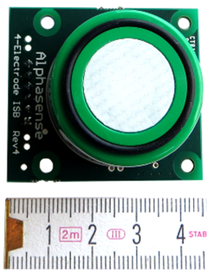


Figure 1.1: **Overview of *OpenSense***: Trams collect measurements of air quality (left), data is collected in private database and forwarded to a public database (middle), data is accessible over a web-interface, which is able to generate simple time-series plots (right).



(a) SGX MiCS-OZ-47 ozone sensor



(b) Alpasense CO-B4 carbon monoxide sensor

(c) Alphasense NO2-B4 nitrogen dioxide sensor

(d) Matter Aerosol MiniDiSC ultrafine particles sensor

Figure 1.2: Air pollutant sensors deployed on top of trams.

four sensors (ozone, carbon monoxide and nitrogen dioxid) are low-cost with a price of around 10 to $100$$ each, the ultrafine particulate matter sensor costs around $7000$$.

The mobile sensors deliver data of high spatial resolution, however an important downside of the inexpensive sensors is their limited precision and reliability. Carbon monoxide sensors e.g. are only equipped with a factory calibration, which converts a measured output voltage linearly to a pollution concentration, which yields limited accuracy. A further problem with using unfiltered data can be seen for the ozone sensors. These sensors need a heat-up time of approximately two hours before they deliver reasonable measurements. Data delivered in this heat-up time is not valid and should therefore not be used to compute statistics. A general problem of the used low-cost sensor is that they drift over their operation time and are susceptible to cross-sensitivities to other pollutants and meteorological influences like temperature or humidity [21]. Therefore it is important to calibrate the measured data samples before they are used for air quality studies to guarantee high quality data.

1.2.2 Data Set

Since the start of measurements in May 2015, over 200 million measurement samples have been collected in a central database (see Table 1.1), which occupy over $250\,\mathrm{GB}$ of disk storage. This data is neither filtered nor calibrated but it is stored as it is delivered by the trams. In addition to

Table 1.1: Overview of measured pollutants (up to 21.10.2016).

Pollutant	# of Measurements	Sampling rate	Time period
Particulate matter	80.3 Mio	5s	05/2012 – 05/2015
Ozone (O_3)	21.3 Mio	30s	05/2012 - ongoing
Carbon-Monoxide (CO)	52.6 Mio	10s	05/2012 - ongoing
Nitrogen-Dioxide (NO_2)	52.6 Mio	10s	05/2012 - ongoing

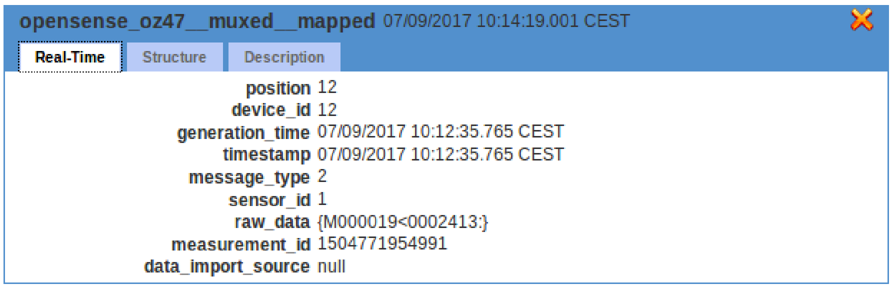
the measured pollutant data of the trams, the database collects also air quality data delivered by static measurement stations operated by official authorities, which is used for calibration. More information about the calibration mechanisms is provided in Chapter 3.

The collected raw data is directly forwarded to a public database accessible over the web-interface. It is neither filtered nor calibrated and also no quality checks are performed. This is problematic, because an observer of the data has no guarantee about its reliability.

An additional challenge is the size of the dataset. When users request arbitrary data over the web-interface, data should be acquired fast and processed efficiently to minimize delays and improve the users experience. This is in the current state not optimally solved, depending on the amount of requested data, users have to wait tens of seconds before they receive an answer from the web-server.

1.2.3 Data Presentation

Currently, the raw data measured by the tram sensors is accessible for the general public over a web-interface [8]. The web-tool can display real time data of the measured pollutants (see screenshot in Figure 1.3) and generate simple time-series plots of the measured data (see screenshot in Figure 1.4).


opensense_oz47__muxed__mapped 07/09/2017 10:14:19.001 CEST

position	12
device_id	12
generation_time	07/09/2017 10:12:35.765 CEST
timestamp	07/09/2017 10:12:35.765 CEST
message_type	2
sensor_id	1
raw_data	{M000019<0002413:}
measurement_id	1504771954991
data_import_source	null

Figure 1.3: Example of raw data live preview on *OpenSense* website.

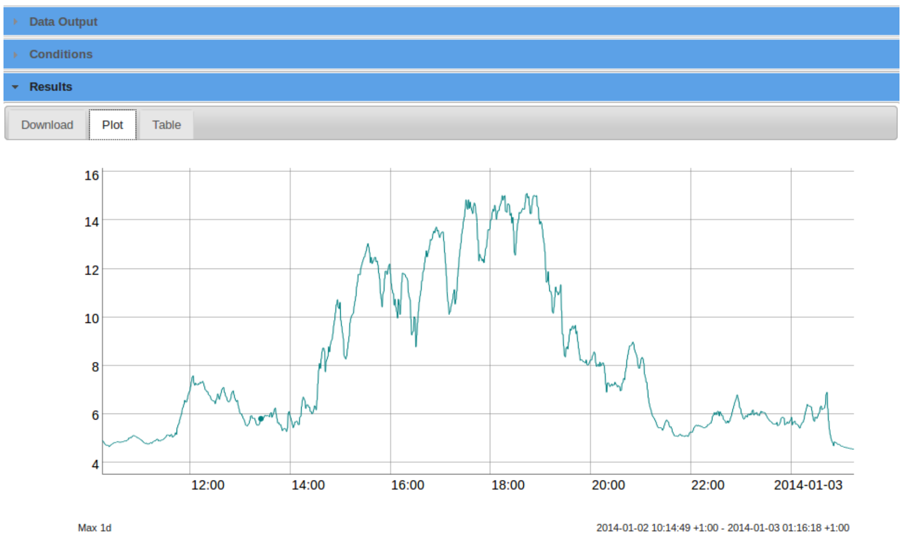


Figure 1.4: Example of raw data time-series plot on *OpenSense* website.

This presentation method is not optimal and of limited use for scientists or the general public. Simple time-series plots hide information about the environment of the tram, as they drive around and change their location and give therefore little insight into the air quality of the city. A more severe problem is that the presented data is used as it is provided by the trams, it is not filtered or calibrated and is therefore of questionable quality. Additionally, the user-interface in its current form can be improved to be more simple and intuitive to understand for people not directly involved in the project.

1.3 Contribution

The goal of this thesis is to improve the previously mentioned flaws in the data-handling part and the presentation module of the *OpenSense* web-tool (see Figure 1.1).

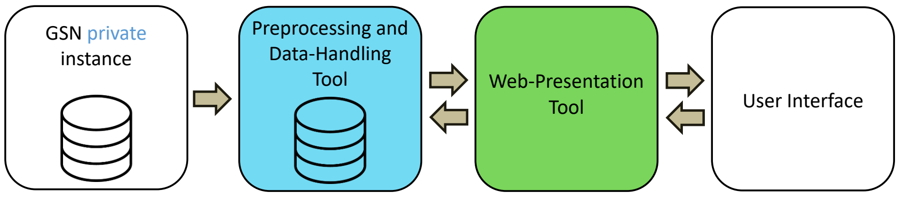


Figure 1.5: Overview of contribution.

The first goal of this thesis is to implement a tool chain that processes the raw measurements delivered by the trams to ensure good quality of the data. This includes

- Removing defective measurements
- Marking possibly incorrect measurement values (outliers)
- Calibrating raw data

Additionally, the tool chain has to perform data summaries to boost up future calculations. The clean up process and the preprocessing of data summaries should run automatized and work with big amounts of data. The calibrated data and the summaries have to be stored and managed in an efficient way such that the interface is able to return the requested data in minimal time.

The tool chain is located between the public data-server that is collecting the raw data delivered by the trams, and a web-presentation tool (see Figure 1.5).

The second goal of this thesis is to setup a web-tool that presents the preprocessed measurement data in a fast, interactive and intuitive way to help people gasp the air quality in Zurich. The web-tool is able to

- plot measurement traces of the trams from different tram tracks in combination with a map showing the tram-track such that the user can gasp the coherence between pollutant values and the geographic location in Zurich,
- provide a heatmap plot onto a city map, showing averaged measurement values for each pollutant in low-resolution cells,
- and provide a heatmap plot of modeled data for ultrafine particles for the whole city.

The presentation tool is inserted after the preprocessing and data-handling tool and is therefore able to acquire calibrated data and preprocessed summaries fast and efficiently (see Figure 1.5).

The following chapters in this thesis are structured as follows:

In chapter 2 related work about *OpenSense* and data filtering is presented. The Chapter 3 describes the implemented filtering mechanism for this thesis, how air quality data is visualized and how environmental data was acquired to build models. In chapter 4 the tool chain to clean up, filter and visualize the raw data automatically is introduced. Chapter 5 presents the evaluation of the results, and in chapter 6, the summary and outlook is presented. Specific information about how to setup and use the tool chain can be found in the Appendix.

Chapter 2

Related Work

This chapter introduces in a first section related work about sensor faults, in a seconds section background information about the *OpenSense* project is provided. In the last section, mechanisms of data visualization are explained.

2.1 Sensor Faults

Inexpensive air quality sensors as used in the *OpenSense* project tend to have a lower precision than expensive high-end sensors used in static measurement stations managed by public authorities. It is therefore important to detect and remove erroneous measurement samples and calibrate the data to ensure a high quality of the dataset before it can be used for air quality studies.

Section 2.1.1 introduces two generic data fault sources in sensor networks. In Section 2.1.2, different data fault types are presented. Section 2.1.3 provides information about error detection methods.

2.1.1 Sensor Network Data Fault Sources

Sensor measurement errors can be classified into two generic main sources according to Elnahrawy and Nath [17]: *systematic errors* and *random errors*. This section introduces these error sources in general, more specific information and examples will be presented in Section 2.1.2.

Systematic errors

Systematic errors arise in a steady manner and have their origin in external influences or alterations of operating conditions. Examples for external influences are climatic influences such as temperature, atmospheric humidity, air pressure or the presence of chemical substances in the atmosphere. Examples for alterations of operating conditions are aging of sensors, changing of the operating frequency or supply voltage due to for example a depleting battery. Systematic errors usually affect measurements over a longer period of time. It exists not only a causality but also a correlation between the measured sensor variable and the disturbance variable.

A systematic error can be corrected in cases where the error leads to a bias, and accuracy but not the precision of the sensors is affected [17][24]. In other cases, e.g. where the measured phenomenon lies outside of the sensitivity of the sensor and it provides faulty data, a correction is not possible.

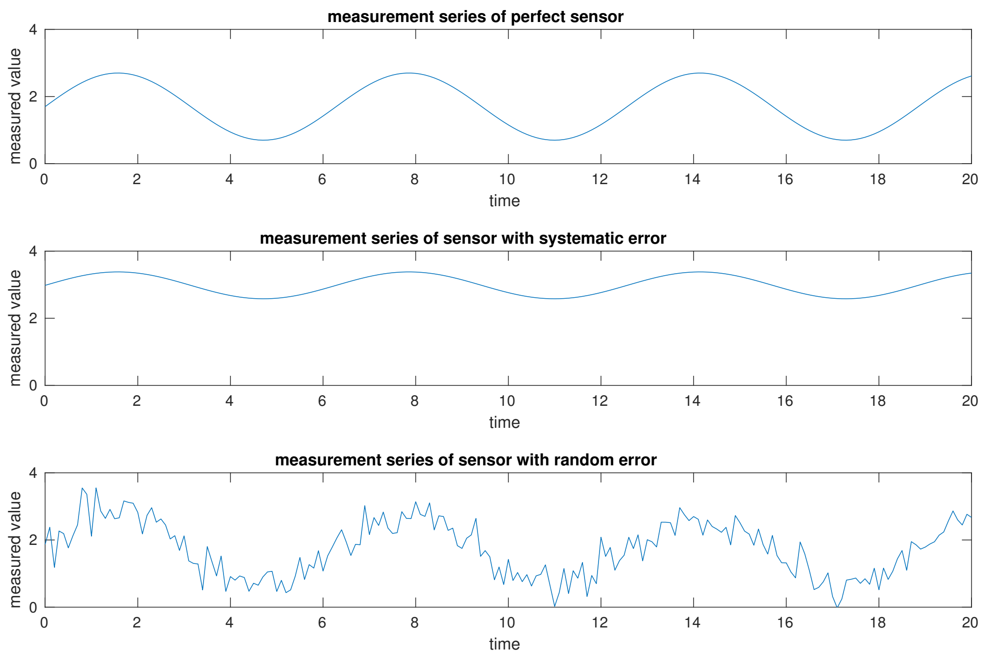


Figure 2.1: Example of a measurement series for an ideal sensor (top), a series with a systematic error (middle) and a series with a random error (bottom).

Random errors

Random errors on contrast to systematic errors are of noisy character and therefore not predictable. Between the measured sensor variable and the disturbance variable exists a causality but no correlation. Random errors may occur due to for example noisy external sources, internal random hardware noise or numerous environmental influences [17].

An example of a systematic and a random error is illustrated in Figure 2.1. The first plot shows the measured signal of a sinusoidal signal of a perfect sensor, the middle plot shows the measurement of a biased sensor, and the third plot shows the measurement of an unbiased sensor but with added white Gaussian noise.

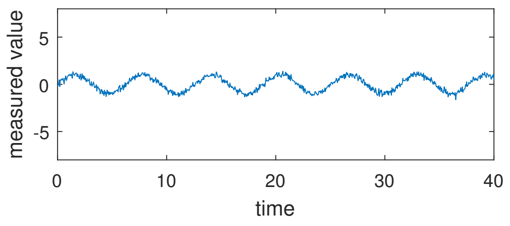
2.1.2 Sensor Network Data Fault Types

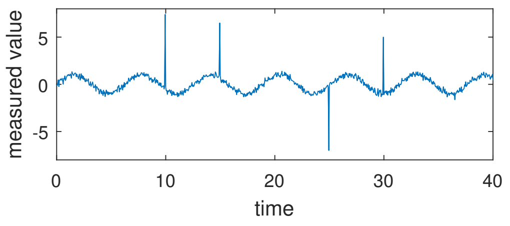
This section presents a more detailed look at sensor faults and their causes. Ni et al. [24] describe two different views how a sensor fault can be classified. In the *data-centric view*, a fault is recognized over a diagnostic approach where the characteristics how the data behaves is analyzed. In the *system-centric view* it is analyzed, in which way a physical malfunction or a fault with a sensor leads to a certain error pattern in the measured output data.

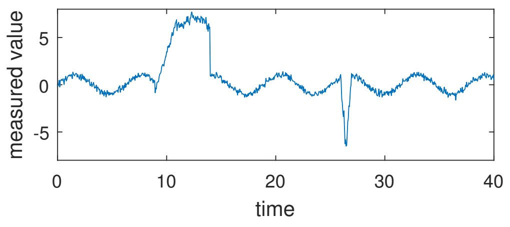
Data-Centric View

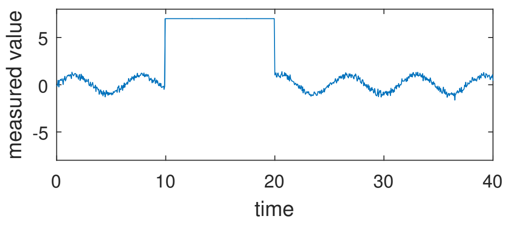
Using the data-centric view, only the raw data samples of a sensor are used to classify a fault. The following sections introduce sensor faults based on this view. The sensor faults are illustrated in Figure 2.2, where in Subfigure 2.2a the sinusoidal example signal without errors is shown.

Outliers Ni et al. define an outlier as *“an isolated sample, in the temporal sense, or a sensor, in the spatial sense, that significantly deviates from the expected temporal or spatial models of the data which are based upon all other observations”* [24, p. 14]. An example of temporal outliers can be seen in Figure 2.2b. To model and detect outliers preferential methods are to examine the distance to other readings (see Sheng et al. [28]) or to examine the gradient (see Ramanathan et al. [25]). Outliers can be the result of a random error, but this does not necessarily have to be


(a) Without errors


(b) Outliers


(c) Spikes


(d) Stuck-at fault

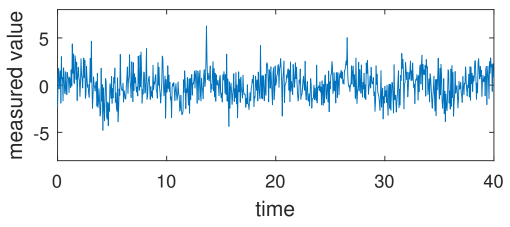

(e) High-noise-or-variance fault

Figure 2.2: Different faults in data-centric view based on a sinusoidal example-base-signal.

the case and depends on the observed phenomenon. If a temperature sensor is recording one sample with a significantly higher temperature than the other samples, a sensor fault is highly likely. On the other side, a light sensor exposed to flashes of a thunderstorm at night will provide data with outliers, where the flashes occurred.

The simplest way to deal with outliers is to simply discard them, because in most cases they do not provide much useful information [29].

Spikes A spike is defined as *"a rate of change much greater than expected over a short period of time which may or may not return to normal afterwards"* [24, p. 15]. In contrast to an outlier, a spike is a consecutive set of data samples instead of only one significantly deviating sample. An example of a measurement series containing spikes is shown in Figure 2.2c. The model to classify spikes must be chosen carefully as Mourad and Bertrand-Krajewski recommend [22], depending on the observed physical phenomenon. The measured data from a humidity sensor for example is not expected to have quick changes, the measured data from a wind velocity sensor on the other side is likely to have large and quick changes when exposed to squalls.

Stuck-at Fault A stuck-at fault is a *"a series of data values that experiences zero or almost zero variation for a period of time greater than expected"* [24, p. 16]. The main characteristics of this fault is its diminished or inexistent variation. In order for the error to be detectable, the correct data around the error must vary. An example of a stuck-at fault is shown in Figure 2.2d. A stuck value does not always mean that the measured sensor samples are erroneous, the measured phenomenon and the precision of the sensor have to be taken into account. A light sensor with limited precision will only deliver zero-samples after the sunset without external light sources present. To distinguish if the measurements are erroneous or not, an effective way is to examine spatial correlation between multiple sensors observing the same phenomenon [24].

High Noise or Variance Some noise is usually expected in sensor measurements, but an increased signal to noise ratio can be caused by a sensor problem, e.g. a hardware failure or dying batteries. Ni et al. define noise faults as *"sensor data exhibiting an unexpectedly high amount of variation"* [24, p. 18]. An example of a high-noise-error can be seen in Figure 2.2e. To develop models able to detect high-noise-or-variance-errors, the expected variance of the sensor has to be taken into account as well as the expected variance of the measured environment variable. To examine if a sensor is faulty or if the variance is caused by the measured environment variable, it is recommended to compare the variability of close by sensors observing the same phenomenon.

System-Centric View

In contrast to the data-centric view, the system-centric view does not identify a fault by examining the measured samples, but examines a physical malfunction, a fault or a condition with a sensor which will result in certain error patterns in the measurement output[24].

Calibration Faults Calibration faults result in a biased measurement output. If a measurement series is only affected by a calibration fault, the precision is not impaired, but the accuracy [24]. In the following formulas, $g(t)$ describes the actual observed phenomenon, $f(t)$ the sensor output of the observation.
Ni et al. define three types of calibration faults [24]:

- An **offset fault** leads to the effect that measured samples differ a constant amount from the actual phenomenon. The sensor output can be described as

$$f(t) = g(t) + C_{offset}$$

  where $C_{offset} \in \mathbb{R}$.

- A **gain fault** leads to an increased change in the sensor output. If the physical phenomenon changes by an amount Δ, the measured value will change by $C_{gain} \cdot \Delta$ (where $C_{gain} \in \mathbb{R}_{>0}$ is a constant). The sensor output can therefore be described as

$$f(t) = C_{gain} \cdot g(t)$$

- A **drift fault** occurs when an initially calibrated sensor changes its properties over time, the source of these changes can be a result of e.g. sensor aging or enduring environmental changes (for example temperature differences in summer and winter). The measured sensor output can be described as

$$f(t) = C_{gain}(t) \cdot g(t) + C_{offset}(t)$$

  where C_{gain} and C_{offset} are no longer constants but functions over time.

Ni et al. [24] state that the detection and correction of calibration errors is difficult without having access to ground truth, because it is a priory not clear if a drift in sensor measurements over time is a drift fault or a variation of the measured phenomenon. A second problem is, that this type of fault can not be detected by simply examining the raw data delivered by the sensor, because the data does not exhibit error patterns in a data-centric view. If a reliable calibration formula to correct offset- and/or gain-faults can be developed, Ni et al. suggest to correct the data instead of discarding it because calibration errors usually only affect accuracy, not precision.

Hardware and System Failures Ni et al. [24] examine also failures of the hardware, circuit connections or other components. Examples of this type of failure are:

- **Connection failures** due to loosen cables, corroded conductors or short circuits. These failures are often caused by environmental conditions, e.g. cables can get loose due to long enduring vibrations and corrosion and shorted circuits can have their cause in humidity entering the sensor. Sensors affected by this type of error need to be replaced or repaired. Examinations showed, that an indication for this type of faults are unexpectedly high or low measurement samples. It is usually difficult to extract useful data of measurement series affected by connection failures, therefore it is suggested to discard affected data.

- A **decreased supply voltage** caused by a dying battery can lead to several effects in the measured data samples, for example spikes, stuck-at-faults or noisy data as described in Section 2.1.2. This error can be easily detected, if the supply voltage is monitored.

Environment out of Range Sensors are constructed to measure an environmental parameter in a certain range. An environment-out-of-range error occurs, when the observed environmental variable exceeds or undercuts the borders of the dimensioned range. This type of fault can be detected by comparing the measured values with the threshold given in the data sheet of a sensor. If the received values lie outside these borders, the confidence of the data must be questioned. A typical observable behavior at the sensitivity borders are stuck-at-faults, described in Section 2.1.2.

2.1.3 Error Detection Methods

Section 2.1.2 introduced different sensor fault types, in this section background information about detection methods of sensor data faults is provided. This topic was examined by Sharma et al. [27]. They make use of the *data-centric view* as introduced by Ni et al. to compare measurement series of different sensors to identify sensor faults. They focus on three different fault models, which are defined similarly as by Ni et al. [24] described in Section 2.1.2: *constant faults* (similar to *stuck-at faults*), *short faults* (generic term for *outliers* and *spikes*) and *noise faults* (identical to *high-noise-or-variance error*).

To detect these faults, they introduced four classes of detection methods for sensor faults [27, p. 3]:

- Rule-based methods,
- estimation methods,
- time-series-analyses-based methods,
- and learning based methods.

These four different error detection methods use different approaches and have different advantages and disadvantages regarding accuracy and robustness. For detailed information, the reader is referred to the paper [27].

2.2 OpenSense Project

The *OpenSense* project was introduced in 2011 with the goal to collect a dataset of air quality measurements in Zurich with high spatial resolution [20]. Low-cost air pollution sensors are integrated into compact air pollution monitoring stations, which are deployed on top of ten trams. The sensors deliver daily megabytes of air quality data, which are stored in a central database.

2.2.1 Dataset

The sensors on top of the trams collect data from the pollutants ozone, carbon monoxide, nitrogen-dioxide and particular matter. Other environmental variables measured are temperature and humidity. All these sensor readings are stored in combination with a time stamp and the GPS position [20]. Since the start of measurements in May 2015, over 200 million measurement samples have been collected. Table 2.1 provides an overview of the collected data of air pollutants by the mobile sensors.

In addition to the data delivered by the mobile sensors, the database collects also air quality data delivered by two static measurement stations operated by official authorities. This data is of high quality and is therefore used as ground-truth. The static measurement stations are located near a tram track and is used to calibrate trams that pass by.

Table 2.1: Overview of measured pollutants (up to 21.10.2016).

Pollutant	# of Measurements	Sampling rate	Time period
Particulate matter	80.3 Mio	5s	05/2012 – 05/2015
Ozone (O_3)	21.3 Mio	30s	05/2012 - ongoing
Carbon Monoxide (CO)	52.6 Mio	10s	05/2012 - ongoing
Nitrogen-Dioxide (NO_2)	52.6 Mio	10s	05/2012 - ongoing

2.2.2 Calibration

To calibrate the measured sensor data, Saukh et al. [26] applied a mechanism called *multi-hop sensor calibration*. This mechanism leverages the property, that trams periodically drive past a static reference station and drive past other trams. When sensors are in close vicinity, it is assumed that the measured values of different sensors observing the same phenomenon are correlating. Temporally and spatially close pairs of measurement values of two sensors are defined as a *rendezvous* (or *checkpoint*). A set of checkpoints between a calibrated and an uncalibrated sensor can be used for calibration. In a first step, checkpoints between trams and a reference station are determined, with whose help the passing by trams can be calibrated because the reference stations are expected to deliver ground-truth-data of high reliability. Since not all trams have common checkpoints with a reference station, checkpoints between already calibrated trams and uncalibrated trams can be determined to recursively calibrate all trams. The name *multi-hop sensor calibration* originates from the property, that sensors may be calibrated over multiple hops. However, this leads to the problem of error propagation. If a sensor is miscalibrated, the error propagates to the next sensors and can accumulate over multiple hops. Saukh et al. examined this problem and conclude, that error accumulation over multiple hops can be reduced, if in the first hop *Ordinary Least Squares Regression* is used and in all following hops *Geometric Mean Regression*.

2.3 Visualization of Air Pollution Data

As described in Section 2.2.1, during the operating time of the *OpenSense* project over 200 million measurement samples have been collected. The measurements must be summarized and presented in abstract form to allow this large amount of raw data to be interpreted. This section presents related work about data visualization of sensor networks.

Vizzly

Keller et al. [19] developed a visualization tool called *Vizzly*. *Vizzly* is a middleware which allows to browse data of large sensor networks in an interactive way. It is able to display map- and line-plots. To visualize small datasets containing a few hundred measurement samples a general approach is to send the raw data to the client who renders them after reception. But this is not feasible if millions of data-points are part of the visualized plot because the transmitted data would be too big and the client would have to perform heavy processing to display the data. The goal of *Vizzly* is to react to user interactions efficiently and provide new data when user requests are adapted. Efficiency is achieved over precomputation of spatio-temporal aggregated data and caching techniques to have fast access to these data. *Vizzly* is used to display data of sensor networks of the *PermaSense*[1] and *OpenSense*[2] project. Evaluations showed, that generating output takes typically less than one second.

Map Generation

The sensor data provided by trams only covers locations where the trams operate, i.e. different tram tracks. Hasenfratz et al. [18] developed land-use regression (LUR) models which allowed

[1] http://data.permasense.ch
[2] http://data.opensense.ethz.ch

them to generate pollution maps for ultrafine particles for the whole city of Zurich with a spatial resolution of $100m \times 100m$. UFP-data is not monitored by the measurement stations of the authorities in Zurich, therefore no ground-truth data is available. To assess high quality of the measurement data, Hasenfratz et al. perform a calibration in a first step. In a second step, unreliable and defective samples are removed. The quality of the data is validated (among others) by analyzing statistical distribution of the measured values and the baseline signal of all devices. The cleaned up data is finally used to model the pollution concentration for all places in Zurich, making use of LUR models. The LUR models use 12 different explanatory variables to build a Generalized Additive Model that is able to extrapolate pollution concentration values where no measurements are collected. Hasenfratz et al. state that this approach works reliable with data of minimum one week. To generate UFP-maps with higher temporal resolution, they propose to add historical measurements with similar metadata for computing the model. With this additional approach they are able to generate semi-daily UFP-maps with sufficient accuracy.

Mueller et al. [23] examined a similar problem for particle number concentrations in Zurich, they generated models with temporal resolutions of 30 minutes and spatial resolutions of $10m \times 10m$.

Chapter 3

Methods

This chapter presents methods and tools needed for the toolchain described in Chapter 4. Section 3.1 presents the mechanisms of cleaning and filtering raw data. Section 3.2 presents, how data is visualized. Certain visualizations are based on environmental data independent from the tram measurements, the acquisition of this data is presented in Section 3.3.

3.1 Data Cleaning and Filtering

To ensure high quality of the data-set, the raw data delivered by the trams has to be filtered and calibrated. This section presents the used methods and algorithms. The filtering and calibration mechanisms vary for different pollutant groups. Section 3.1.1 presents universal filtering methods, the subsequent Sections explain the clean up process for ozone and carbon monoxide (filtering in Section 3.1.2, calibration in Section 3.1.3) and Section 3.1.4 explains the clean up process for measurements of ultrafine particles.

3.1.1 Filtering Methods

Temporal Filters

There exist various reasons why collected sensor data in a certain timeframe is not suitable for statistical summaries and needs to be discarded. Examples are:

- The sensor box is dismounted from the tram for maintenance or testing and is running inside a laboratory. Thus, the recordings are not collected in the targeted environment.
- Manual examination of the sensor data suggests that the sensor is broken or faulty with high probability.
- A sensor has reached the expected lifespan and future measurements have to be discarded for reliability reasons.

In our toolchain the sensor IDs and the desired timespan of faulty measurements are stored in a configuration file.

Spatial Filters

Data of certain places must not be used, because the places lie not within the desired region to monitor. For the *OpenSense* project these places are the tram depots where the trams return after service. When the trams return to the depot, the sensors keep running as long as the trams are powered and keep delivering air quality measurements. Since these measurements hail from inside a building, they do not directly correspond to the air quality outside. To preclude an influence on the calibration and on the model generation, all measurements from inside these depots are removed. The coordinates of the tram-depots to remove are stored in a configuration

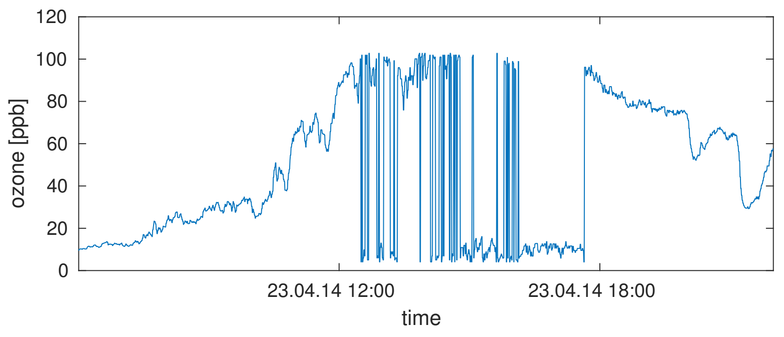


Figure 3.1: Unfiltered and uncalibrated data delivered by the ozone sensor of tram 8 with defective measurements.

file with thresholds on how much data around these locations should be removed. In the used configuration for this thesis, the threshold is set to $100\,\mathrm{m}$. In addition to data collected inside the tram-depots, data collected on the roof of the ETZ building of ETH is removed, where the sensors are deployed during a testing-phase.

Incomplete Measurements

Incomplete measurements are not usable to compute statistics and are therefore removed if one or more parameters are missing. These are *NULL* records in the MySQL database.

Inaccurate GPS position

The GPS receiver delivers a measure for the quality of the measured geometric position, the *horizontal dilution of precision* (HDOP). The smaller this value gets, the better the quality of the received signal is. To assure a high geometric position accuracy of the measurements, all records with a HDOP value bigger than 3 are deleted, as suggested by Hasenfratz et al. [18, p. 3].

3.1.2 Filtering of Ozone and Carbon Monoxide Data

Section 3.1.1 introduces filtering methods applied to all pollutants, this section explains additional filtering methods used to clean up the raw data of ozone and carbon monoxide data.

Boot-Up Phase Removal

Certain sensors do not deliver meaningful data right after boot-up and must run for a given timespan before the sensors deliver reasonable values. In the *OpenSense* project the ozone sensor is affected by this problem. To filter these records, two thresholds are defined:

- $T_{cooldown}$ defines the timespan, after which a previously powered on sensor has again to go through the boot-up process when it is turned off.
- T_{bootup} defines the timespan after the sensor is turned on, in which the measurements are not valid.

Whenever the time difference ΔT between two consecutively recorded samples is bigger than $T_{cooldown}$, all samples within the timespan T_{bootup} after the seconds sample are removed.

Marking Possibly Defective Measurements

The low-cost sensors deliver sometimes unreliable measurements for a period of time. This problem can be seen in Figure 3.1. This example visualizes the output voltage of the ozone sensor during the day on 23.04.2014. The measured concentration increases during the morning to around $100\,\mathrm{ppb}$. After 12 o'clock the measured sensor values drop to very low values of around $10\,\mathrm{ppb}$ for a few samples and rise again to $100\,\mathrm{ppb}$ almost instantly. It is likely that these measurements are faulty, because they are taken in the middle of the city and the tram is always in a similar environment. To detect measurements which could be outliers caused by sensor malfunction, a Hampel filter is applied to remove outliers in a copy the data. The processed data is then compared to the original measurements and all samples where the values differ are marked as possibly defective.

A Hampel filter [6] is a windowed filter that is able to detect and, if needed, correct outliers. For each sample S_N ($N \in \mathbb{N}_0$) in a time series of measurements, the algorithm takes X ($X \in \mathbb{N}_0$) surrounding consecutively measured values before and after the sample S_N into account, this results in the set

$$S = \{S_{N-X}, ..., S_N, ..., S_{N+X}\}.$$

In a first step, the local median MED_N for the sample S_N is computed, which is defined as

$$MED_N = \mathrm{median}(S_{N-X}, ..., S_N, ..., S_{N+X}).$$

After that, it estimates the standard deviation, defined as

$$ESTD_N = \frac{1}{\sqrt{2}\mathrm{erfc}^{-1}(1/2)} \cdot \mathrm{median}(|S_{N-X} - MED_N|, ..., |S_N - MED_N|, ..., |S_{N+X} - MED_N|\}).$$

If for a sample S_N the condition

$$|S_N - MED_N| > 3 \cdot ESTD_N$$

holds, then the Hampel filter identifies S_N as an outlier and replaces its value to

$$S_N = MED_N.$$

For this thesis, the window delimiter X is set to 100 samples.

3.1.3 Calibration of Ozone and Carbon Monoxide Data

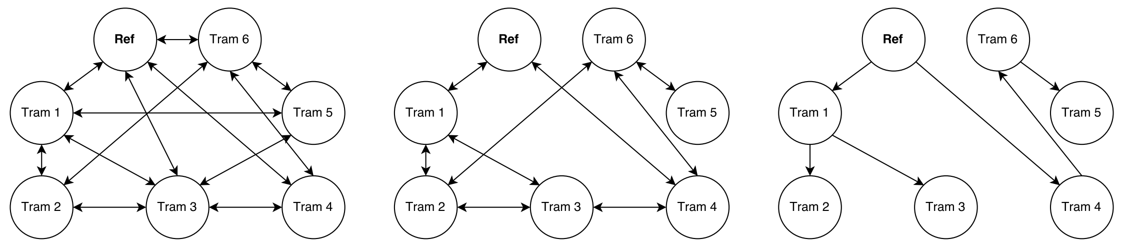
The calibration of ozone and carbon monoxide data is performed with the multi-hop calibration method introduced by Saukh et al. [26]. This calibration method is briefly summarized in this section. For a full description of the procedure, the reader is advised to consult [26].

Checkpoint Computation

In a first step, *rendezvous* (or *checkpoints*) between all sensors are computed, this can be between a reference station and a tram or between two trams. Saukh et al. define a rendezvous as a *"set of spatially and temporally close pairs of measurements"* between two sensors [26, p. 2]. One element of this set is referred to as a *calibration pair* (or *checkpoint*). Checkpoints are defined by a spatial (Δd) and a temporal (Δt) constraint variable, which define how close in time and space two measurements must at least be to be in the same checkpoint. Saukh et al. state, that plausible choices for these variables are $\Delta t = 5\,\mathrm{min}$ and $\Delta d = 50\,\mathrm{m}$ [26, p. 9]. They are used unaltered for this thesis.

Calibration Path Selection

To ensure a good quality of the calibration, the paths between the devices have to be selected carefully. Figure 3.2 presents an example of the procedure of selecting calibration paths. In this example, a sensor of a reference station and sensors of six trams are part of the topology. The algorithm to select a calibration path is implemented as follows:



(a) Sensors with common checkpoints.

(b) Sensors with common checkpoints that satisfy restrictions.

(c) Final multi-hop calibration path.

Figure 3.2: Example of procedure of calibration path selection for multi-hop calibration algorithm.

1. In a first step, all sensor with common checkpoints are connected with an edge (double arrow). This is illustrated in Figure 3.2a
2. However, to reduce calibration errors, common checkpoints between two sensors have to satisfy certain quality requirements (see next section) to lower the risk of increased calibration errors. In Figure 3.2b, all edges are removed where the quality requirements for rendezvous are not satisfied.
3. In the last step a tree with minimal depth is extracted with the reference station as root. This step is illustrated in Figure 3.2c.

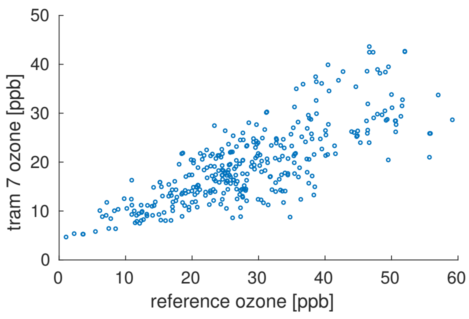
Quality Requirements for Checkpoints

The checkpoints between two sensors have to satisfy certain quality requirements to lower the risk of severe calibration errors:

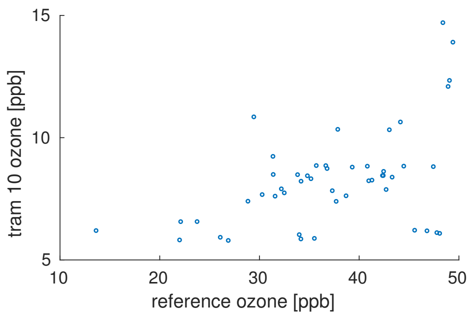
- The **number of checkpoints** of two sensors can notably impact the calibration accuracy, i.e. a small checkpoint set usually yields higher errors than big sets. Single measurements of sensors are affected by random errors, and if only a handful of checkpoints are used for calibrating a sensor, these errors have a high influence, therefore a threshold has to be set for the required minimal number of checkpoints between two sensors such that one is allowed to calibrate the other. The bigger the number of checkpoints is, the lower the influence of a single random-error gets. But on the other side, if the threshold is too big, single sensors could get isolated and not be calibrated.
- A second important quality feature is the **correlation** between the measurement values of checkpoints between two sensors. If the correlation is too low, it is an indicator that at least one of the involved sensors is affected by intense random errors. This leads again to a tradeoff: If the requested correlation is chosen too high, sensors can again lose all their rendezvous with other sensors and can not be calibrated. If the desired correlation value is too small, noisy sensors are not excluded from calibration paths and a big calibration error can be passed over multiple hops. Saukh et al. set the correlation threshold to 0.5 and drop all rendezvous with a lower correlation.

In this thesis, different thresholds are tested to build the calibration paths. For the number of checkpoints between two neighboring sensors, the values 30, 50 and 100 are used. For the correlation between the measurement values of checkpoints, the value -1, 0.3, 0.4, 0.5 and 0.6 are used. This leads to 15 different configuration possibilities. The standard configuration for the web-tool is set to 100 for the minimal number of checkpoints between two sensors, and to a minimal correlation of 0.5.

Two examples of checkpoints between a reference station and a tram are shown in Figure 3.3. Figure 3.3a shows the checkpoints between the reference station and the tram with ID 7 and Figure 3.3b shows the checkpoints between the reference station and the tram with ID 10. The comparison between these two plots shows that the tram with ID 7 in the first plot is more suited to be used as a first calibration hop, the checkpoints are numerous (338) and have a high correlation coefficient (0.785). The tram with ID 10 on the other side delivers fewer samples (49) than tram 7 and the correlation coefficient is much lower (0.4933).



(a) Checkpoints between reference station and tram 7. Number of checkpoints: 338, correlation coefficient: 0.785.



(b) Checkpoints between reference station and tram 10. Number of checkpoints: 49, correlation coefficient: 0.493.

Figure 3.3: Checkpoints between reference station and two tram sensors measuring ozone between 01.05.2014 and 01.06.2014.

Calibration Window Length

Another important factor which influences the quality of the calibration is the time period from which the checkpoints are computed and used for calibration (later referred to as **calibration window length**). The bias of sensors can be affected by environmental parameters such as temperature or air humidity which change significantly over the year. Additionally, single sensors can be affected by drift faults and change their bias over time. Therefore, a reasonable calibration window size has to be selected. On one side, the calibration window length must not be too big, otherwise effects as drift faults have a negative impact on the calibration. On the other side, the calibration widow length must not be too small, otherwise the number of computed checkpoints is too low for a reasonable configuration.

In this thesis, the calibration window length is set to three weeks. To calibrate the whole dataset, a windowed approach is used: To calibrate data within a certain week, the data of the previous and the following week is also taken into account. To calibrate data of week W_N $(N \in \mathbb{N}_0)$, the checkpoints of the weeks W_{N-1}, W_N and W_{N+1} are used.

Multi-Hop Calibration

The previous sections explains, how checkpoints between sensors are computed and how a calibration path is selected. This section explains the calibration mechanism.

Checkpoints between an uncalibrated sensor and a calibrated one can be used for calibration with line fitting methods where two calibration parameters α and β are computed, such that an uncalibrated sensor value x gets corrected to the value $\hat{x}$:

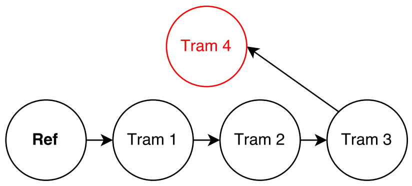
$$\hat{x} = \alpha + \beta x. \tag{3.1}$$

Saukh et al. use for the first hop between a static station delivering ground-truth data and an uncalibrated tram ordinary least squares regression (*OLS*). To calibrate an uncalibrated tram-sensor with the measurements of a calibrated tram-sensor, they use geometric mean regression (*GMR*). This procedure is used unaltered in this thesis.

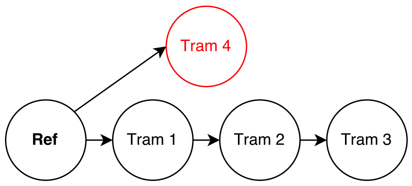
Discarded Approaches

In addition to the previously mentioned filtering and calibration methods, other mechanisms are explored which are not improving the calibration results and are therefore not implemented in the final toolchain. These methods are briefly described in this section.

Smoothing Data before Calibration In a first approach is evaluated if data smoothing improves the quality of the calibration. Smoothing data summarizes multiple data samples in a

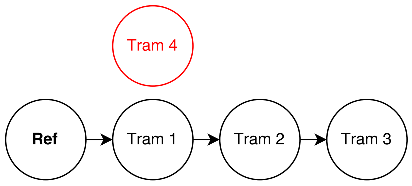


(a) Calibration path without data smoothing.

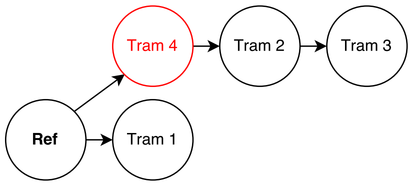


(b) Calibration path with data smoothing.

Figure 3.4: Example of smoothing data with advantageous effect on calibration error.



(a) Calibration path without data smoothing.



(b) Calibration path with data smoothing.

Figure 3.5: Example of smoothing data with disadvantageous effect on calibration error.

windowed approach. This leads to the effect that random errors with a noisy character on a single sample are reduced because multiple errors get summarized. To smooth the data, the following filters are tested:

- Moving average filter
- Moving mean filter
- Savitzky-Golay filter
- Kolmogorov-Zurbenko filter
- Hampel filter

Evaluations for different configurations show that this approach must be used with caution. Smoothing the data can lead to the effect that some checkpoints, previously influenced by severe random errors, get closer to the values of the other checkpoints. This can lead to an improved correlation coefficient between sensors, which in return can lead to positive or negative effects.

Figure 3.4 shows an example of a positive effect. Without applying data smoothing in Figure 3.4a, the tram with ID 4 is attached to the end of the calibration path. When smoothing is applied in Figure 3.4b, the correlation coefficient between the checkpoints of the reference station and the tram gets bigger and the tram can directly be calibrated with the measurements of the reference station. Tram 4 suffers now not anymore from the error accumulated over multiple hops.

An example of a negative effect can be seen in Figure 3.5. Without applying smoothing in Figure 3.5a, the tram with ID 4 can not be attached to any calibration path, its measurements are not calibrated and discarded. When smoothing is applied in Figure 3.5b, the tram can be attached to the measurement station. But the path selection algorithm has set now tram 2 and tram 3 as children of tram 4 instead of tram 1, which can lead to an increased error because the checkpoints of tram 4 are of lower quality than those of tram 1.

To actually benefit from smoothing and filtering algorithms, the sensors have to be modeled accurately to detect errors reliably. Implementing these algorithms goes beyond the scope of this thesis, therefore this approach is rejected.

Binning Checkpoints The measurement values of the checkpoints are in most cases not equally distributed over the measurement scope of a sensor. Figure 3.3b shows an example of this phenomenon. Between the measurement values 10 and 20, the reference station exhibits only one checkpoint. Between the measurement values 40 and 50, the reference station exhibits 20 checkpoints. This leads to the problem, that the sensor scope between 40 and 50 has a much higher influence on the calibration than the scope between 10 and 20. Such that all intervals of the scope would influence the calibration equally, our approach is to bin the data into equally distributed bins and calculate perform the calibration with the averaged values of the bins. This does not lead to an improvement in the calibration error, therefore this approach is rejected.

3.1.4 Clean up Process of Ultrafine Particle Data

Ultrafine particle data can not be calibrated in the same way as the measurements of carbon monoxide and ozone as described in Section 3.1.3, because the static measurement stations do not monitor ultrafine particle data, i.e. ground-truth is missing. To calibrate the measurements of the ultrafine particle sensors, the methods of Hasenfratz et al. [18, p. 3] are used. The *MiniDiSC* devices which measure data about ultrafine particles go periodically into a self-calibration mode to measure their null-offsets. The null-offsets are later used in offline computations to adjust the offset of the measurements. After correcting the offset, they remove records with inaccurate GPS positions (as described in Subection 3.1.1). The *MiniDiSC* devices monitor different status variables during their operation time, e.g. if the air-flow in the sensor is high enough to deliver reasonable measurement values. These parameters allow inferences to be made about the correct operation of the sensors. If the value of a status parameter suggests that a problem occurred, the affected measurement samples are deleted. The *MiniDiSCs* need a heat-up time of one hour after booting up where they deliver inaccurate measurements, data collected in this timespan is deleted. For more detailed information, the reader is advised to consult [18]. All these filters are implemented in the toolchain of this thesis.

3.2 Data Visualization

This section introduces how the cleaned up data is visualized. The goal of the data visualization is to generate different interactive plots that are intuitive to understand for the general public and to help people gasp the air quality in Zurich. To present the data, three plot types are elected: Trace plots which visualize time series of measurements on a tram-track (see Section 3.2.1), heat maps summarizing data for cells in Zurich (see Section 3.2.2) and model-plots for ultrafine particle data (see Section 3.2.3).

3.2.1 Trace Plots

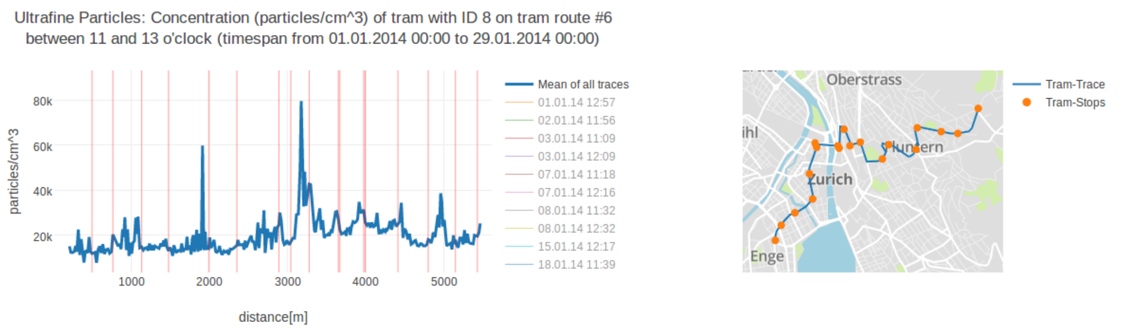


Figure 3.6: Screenshot of a *trace plot* in web-interface.

Series plots are an intuitive way to grasp the air quality. Figure 3.6 shows an example, how data is visualized. The *trace plot* is shown on the left side. The measured pollution (y-axis) is displayed over the traveled distance of the trams on a certain tram track (x-axis). For a better spatial overview the tram-track is plotted onto a map on the right side together with tram-stops. The tram-stops are also marked in the *trace plot* with vertical lines. When hovering over a data point in the trace, the name of the closest tram-stop is shown, together with the traveled distance and the measured pollutant value. By default, only the mean of all extracted tram-traces is shown in the *trace plot*, but the user can dynamically enable the visibility of single traces by clicking on the corresponding label in the legend of the *trace plot*.

In the interface the user can select which tram trace and which sensor should be displayed for which time period. Additionally the user can filter for certain daytimes. An algorithm applies the filters to the cleaned up raw data, and extracts all measurement series captured between the start and the end of the selected tram-track. To extract the time series belonging to a certain track, the raw data is compared to a GPS-track of the tram trace. More information about the GPS-tracks is provided in Section 3.3.

3.2.2 Heatmap Summary Plot

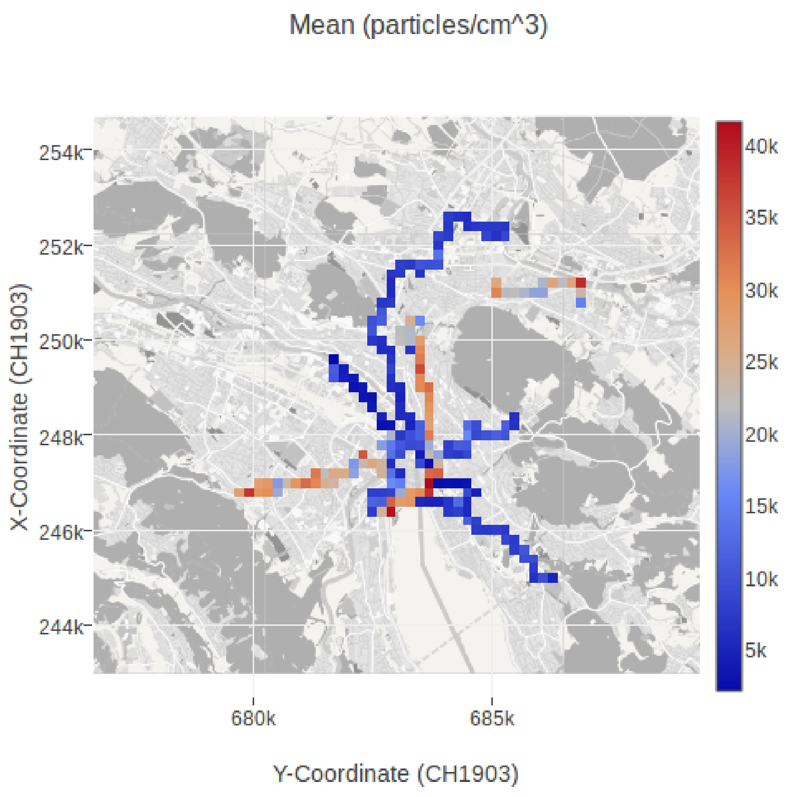


Figure 3.7: Screenshot of *heatmap summary plot* showing the mean concentration of ultrafine particles at different locations in the city of Zurich between 01.03.2014 and 08.03.2014.

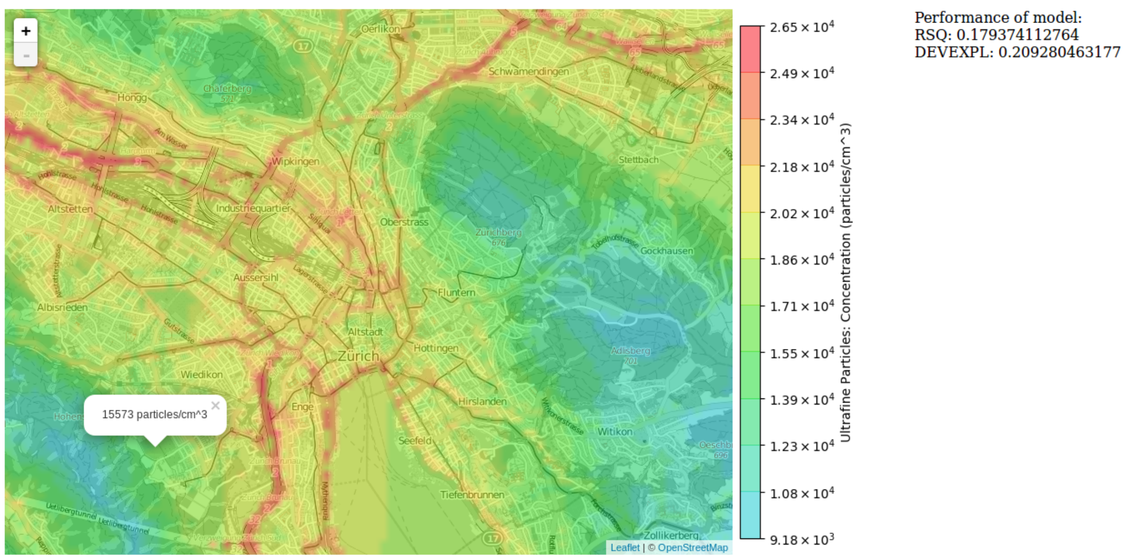
Heatmap plots are a good approach to summarize data and to get a better overview of the pollutant distribution in the city. When the user requests a *heatmap summary plot*, cleaned up raw data of the requested timespan is loaded and summarized for cells with a dimension of $200\,\mathrm{m} \times 200\,\mathrm{m}$. Figure 3.7 displays an example of a *heatmap summary plot*, where the mean of each cell is displayed. The web-tool displays additionally two more plots, for the standard-deviation and the number of samples available for each cell side by side with the mean. The *heatmap summary plot* can be requested for user-defined data up to one month or as a monthly precomputed version implemented in a slide-show.

3.2.3 Ultrafine Particles Map Plot

This section explains the generation of *ultrafine particles map plots*. An example of such a map can be seen in Figure 3.8. The next sections explain, how a model for ultrafine particle data is computed and how the model is visualized.

Model Computation

For ultrafine particles, Hasenfratz et al. [18] develop land-use regression models which allow them to generate pollution maps for the whole city of Zurich with a spatial resolution of $100\,\mathrm{m} \times 100\,\mathrm{m}$ (see also Section 2.3). The model calculation algorithm is also used for this thesis to



```
Performance of model:
RSQ: 0.179374112764
DEVEXPL: 0.209280463177
```

Figure 3.8: Screenshot of *ultrafine particles map plot* for particle concentration between 01.01.2014 00:00 and 08.01.2014 00:00.

visualize modeled pollution data on an interactive map. The procedure is briefly summarized in this section.

In a first step, Hasenfratz et al. collected twelve different land-use datasets of explanatory variables which are used in a later step to build a Generalized Additive Model. In this thesis additional land-use models are generated (see Section 3.3), but the final algorithm is only based on the explanatory variables originally used by Hasenfratz et al.

In order to calculate the model, the requested data is first divided into a grid that covers the whole city of Zurich and whose cells are $100\,\mathrm{m} \times 100\,\mathrm{m}$ in size and for each cell a statistical summary is computed, like mean, standard deviation, median and others. In a second step, the grid cells are extended with statistical information based on the explanatory variables, like traffic data or population density data. Finally a Generalized Additive Model is computed with the statistical summaries and the explanatory variables. Besides the modeled data, the algorithm delivers statistical data to validate the quality of the computed model.

Model Visualization

To visualize the model, a heatmap as an image is generated and overlaid over an *OpenStreetMap* [11] based interactive map. The map is customized with Leaflet [7]. An image generated with the data of the previously described model is embedded onto the map as a transparent overlay. Additionally, a pop-up functionality is implemented which allows a user to click on the map and receive the pollutant load at the clicked position. In addition to the interactive map, a color bar and statistical information of the model quality is shown (see Figure 3.8 right side). The *ultrafine particles map plots* can be requested for user-defined data up to one month or as a monthly precomputed version implemented as a slide show.

3.3 Acquisition of Environmental Data

For this thesis, additional data besides air pollutant measurements has to be acquired. The *trace plots* explained in Section 3.2.1 need GPS data of all tram-tracks, whose acquisition is explained in Section 3.3.1. Metadata can be used to refine models, as described by Hasenfratz et al. [18]. Section 3.3.2 explains, how weather data is acquired that can be used to add metadata to models. In order to create a model for an entire region with data from a few cells, it is important to have access to explanatory variables of the modeled environment. Section 3.3.3 describes additional sources for such explanatory variables.

3.3.1 GPS Tracks of Trams

The GPS information of the tram tracks used for the *trace plots* explained in Section 3.2.1 is manually generated with the the web-tool *ViewRanger* [16] based on data provided by *Google Maps* [5]. The raw data is exported to *gpx*-files and further processed by interpolating the GPS track, such that two consecutive points have a maximal distance of $25\,\mathrm{m}$. A high density of points is needed to accurately project the measured data from the trams onto the tram-track. Additionally, each GPS point is extended with information about the closest tram stop. Information about tram-stops is acquired from the dataset of *OpenStreeMap* [11].

3.3.2 Weather Data

Weather data can be used as metadata, for example to supplement models with data from other periods of time, but similar metadata as described by Hasenfratz et al. [18]. In this thesis, data from two measurement stations (Mythenquai and Tiefenbrunnen) in Zurich is acquired. Information about air temperature, atmospheric pressure, wind direction and speed, air humidity, gust velocity, wind direction, windchill, water temperature in the lake and the dew point is stored. Data from these two stations is available as a historical dataset (since 2011). Data for a third station (Stampfenbachstrasse) can be acquired by polling data from a live-preview web-server. For this station air pressure, temperature and wind-direction and -speed is available.

3.3.3 Land-use Data

Land-use data can be used to generate models that predict pollution values at locations where no measurements are acquired, as described in Section 3.2.3. In this thesis, three different land-use datasets are generated for Zurich: A dataset for building area density, for altitude and for traffic data. This data is an update of the already existing data collected by Hasenfratz et al. [18]. The datasets are introduced in the next sections.

Building-Area Density

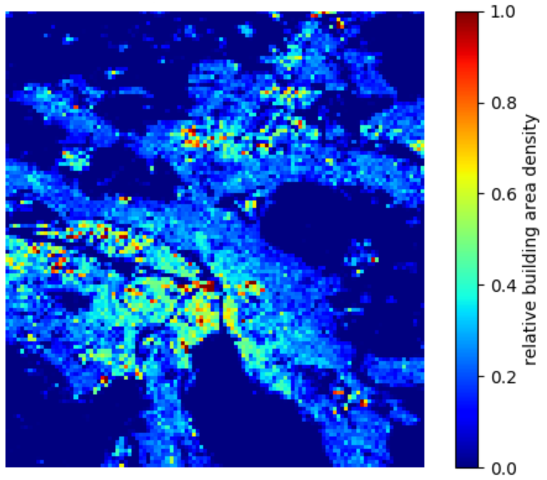
The dataset for building-area density is based on data of *OpenStreetMap* [11]. To generate the model of the building-area density, the city of Zurich is divided into a grid by cells with size $100\,\mathrm{m} \times 100\,\mathrm{m}$ in a first step. In a seconds step, all information about buildings is extracted from the *OpenStreetMap* dataset, buildings are stored as polygons [10]. In a final step, the surface of all polygons is segmented to the individual grid elements. If a polygon lies on several cells, the area is divided proportionally. The generated land-use data for the building density can be seen in Figure 3.9a.

Elevation

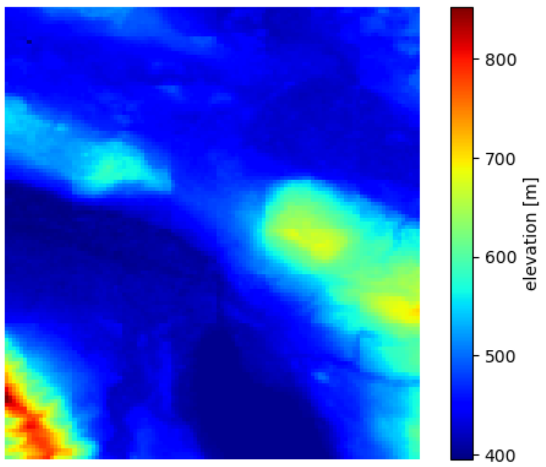
The elevation model is based on data provided by the *Shuttle Radar Topography Mission* (SRTM) [14]. The goal of this mission is to build a global dataset for elevation data. The data is free to download on the webpage of the *United States Geological Survey* (USGS) [4]. The data for Europe is sampled with a sampling rate of around three arc-seconds, which leads to a resolution of around $90\,\mathrm{m}$[3]. Based on this data an elevation dataset for the city of Zurich is created in this thesis. The data is re-sampled to a grid with a cell size of $100\,\mathrm{m} \times 100\,\mathrm{m}$. Figure 3.9b shows the generated land-use dataset for elevation.

Traffic Data

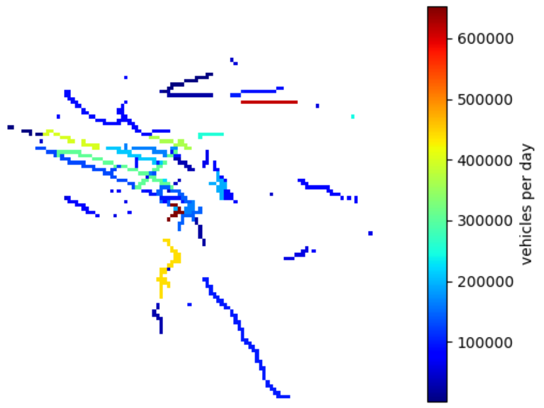
The traffic data model is based on traffic count data provided by official authorities delivered by sensors in the streets that count vehicles [15]. Additionally, the model uses the *Open-StreetMap* [11] database to extract information about the road-system of Zurich. In order to determine the traffic load on a road, measured values of several days are summarized. In a



(a) Building-Density in Zurich.



(b) Elevation in Zurich.



(c) Traffic Model of Zurich.

Figure 3.9: Overview of land-use datasets.

first step, road elements are extracted of the *OpenStreetMap* data and the single segments are connected. In a second step, the sensors are projected onto the street grid based on their GPS coordinates. In a last step, the modeled traffic volume is applied to the whole street when it contains a counting station. A visualization of the model can be seen in Figure 3.9c.

Chapter 4

Design and Implementation

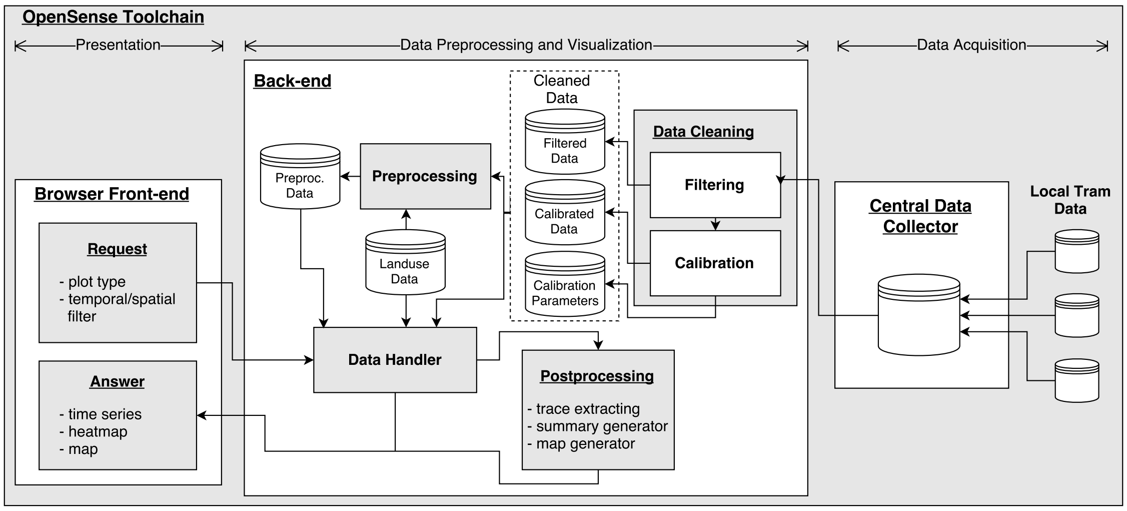


Figure 4.1: Overview of the *OpenSense* toolchain.

This chapter presents the implementation of the automated toolchain, which is able to clean up and calibrate the collected raw data of the trams, calculate data summaries and present the data in an efficient and intuitive way on a web-interface. An overview of the toolchain is given in Figure 4.1. On the right side, the data acquisition part of the toolchain can be seen. As the trams are driving through the city and collect air quality measurements, they store the measurements locally. This data is forwarded to a central database that stores and manages the data of all trams. The data collection is presented in Section 2.2.1. The back-end of the toolchain can be seen in the middle of the diagram. Raw data from the central database is cleaned up by applying filters and calibrating the data. The cleaned up samples and calibration information (depending on pollutant) are stored in new databases shown on the left side of the data cleaning module. This data is used to preprocess data summaries and pollution maps which are stored in a separate database shown on the left side of the preprocessing module. Certain plots require landuse data besides the pollutant measurements, this data is stored in a separate database. The back-end is accessed over a web-interface that allows a user to request plots of data. When the back-end receives a plot-request, a data handler choses the appropriate data (cleaned-up raw data or preprocessed data-summaries). The data is then forwarded either directly to a visualization tool, when preprocessed data-summaries are requested, or via a post-processing module, when custom plots requiring raw data is requested. Finally, the back-end returns the visualized data to the front-end that displays the plot in a web-browser. The following sections present this toolchain in depth.

4.1 Data Cleaning Module

To ensure high quality of the data-set, the raw data stored in the central database has to be filtered and calibrated. Data of different pollutants is cleaned up and calibrated in different ways, as explained in Section 3.1.

Ozone and Carbon Monoxide The pollutants ozone and carbon monoxide are first cleaned up by applying different filters and validating the data as described in Section 3.1. These samples are stored uncalibrated in the database *Filtered Data* shown in Figure 4.1. After the filtering process, calibration parameters are computed with a multi-hop calibration for different thresholds as explained in Section 3.1.3. These calibration parameters are store in a separate database labeled as *Calibration Parameters* in Figure 4.1. When calibrated data for ozone or carbon monoxide is requested, the cleaned samples and calibration parameters are loaded separately and the data is calibrated by the requesting module. This procedure leads to slightly higher computation times contrasted to the case where the data is stored already calibrated, but because 15 different calibration configurations are available as describe in Section 3.1.3, the disk storage overhead can be massively decreased.

Ultrafine Particles Ultrafine particles are filtered and calibrated in one step as described in Section 3.1.4. These cleaned and calibrated measurements are stored in a database labeled as *Calibrated Data* in Figure 4.1.

4.2 Preprocessing and Postprocessing Modules

The preprocessing and postprocessing modules generate visualizations of calibrated data as described in Section 3.2. They acquire the data from the three databases explained in Section 4.1 and if landuse data is needed additionally from the landuse database.

The preprocessing module precomputes *heatmap summary plots* and *ultrafine particles map plots* (see Section 3.2.2 and 3.2.3) on a monthly basis. The plots are stored in a database labeled as *Preproc. Data* in Figure 4.1. Loading precomputed plots is much faster than computing it from scratch, as only the plot has to be served over the network and all calculations are omitted.

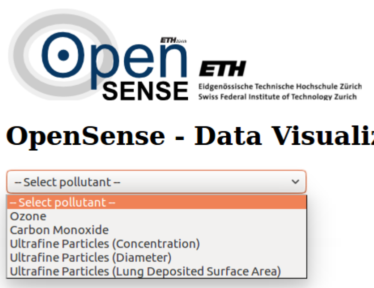
The postprocessing module is called, when a user requests any plot described in Section 3.2 which is not precomputed. This is the case for *heatmap summary plots* and *ultrafine particles map plots* for other than monthly resolutions and for *trace plots* in general.

The pre- and postprocessing module use the following libraries to generate the visualizations:

- Plotly [13] for the series plot of the *trace plots* (see Figure 3.6, left side) and for the *heatmap summary plots* (see Figure 3.7),
- the Mapbox-plug-in for Plotly [12] to generate map of the *trace plots* (see Figure 3.6, right side),
- and the JavaScript library of Leaflet [7] to generate the *ultrafine particles map plot*.

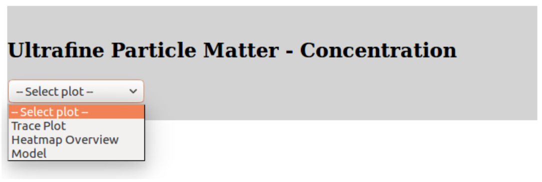
4.3 Data Handler

The data handler processes the requests sent to the web-server and checks if a preprocessed plot can be served. If this is possible, the data handler loads the plot from the database containing preprocessed data (labeled as *Preproc. Data* in Figure 4.1) and returns it to the user over the network. If a plot is requested not readily available, the data handler acquires the needed sensor-data and if necessary landuse data and returns it to the postprocessing module that generates the plot and returns it to the user.

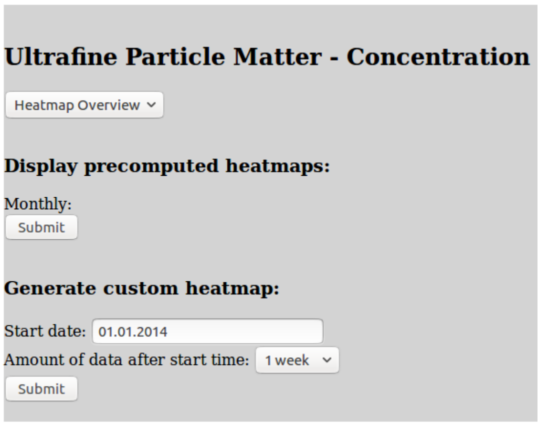

OpenSense - Data Visualization Tool

Open SENSE
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
– Select pollutant –
– Select pollutant –
Ozone
Carbon Monoxide
Ultrafine Particles (Concentration)
Ultrafine Particles (Diameter)
Ultrafine Particles (Lung Deposited Surface Area)

(a) Main page of web-interface with drop-down menu to select desired pollutant.


Ultrafine Particle Matter - Concentration

(b) Drop-down menu for selecting the plot type.


Ultrafine Particle Matter - Concentration

Heatmap Overview

Display precomputed heatmaps:

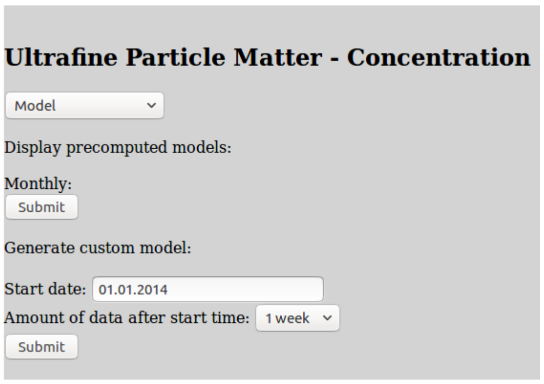
Monthly:

Generate custom heatmap:

Start date: 01.01.2014

Amount of data after start time: 1 week

(c) Interface for requesting a *heatmap summary plot*.


Ultrafine Particle Matter - Concentration

Model

Display precomputed models:

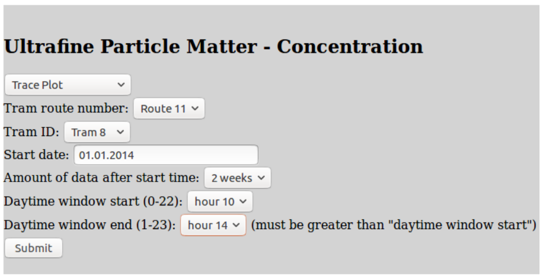
Monthly:

Generate custom model:

Start date: 01.01.2014

Amount of data after start time: 1 week

(d) Interface for requesting an *ultrafine particles map plot*.


Ultrafine Particle Matter - Concentration

Trace Plot

Tram route number: Route 11

Tram ID: Tram 8

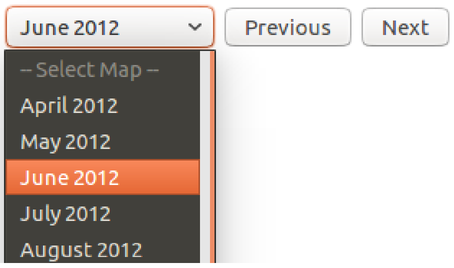
Start date: 01.01.2014

Amount of data after start time: 2 weeks

Daytime window start (0-22): hour 10

Daytime window end (1-23): hour 14 (must be greater than "daytime window start")

(e) Interface for requesting a *trace plot*.



(f) Interface for changing the precomputed plot.

Figure 4.2: Overview of web-interface.

4.4 Browser Front-end

In the browser front-end, a user can specify a plot request. Figure 4.2 provides an overview of the web-interface. In a first step, the user selects the desired pollutant shown in Figure 4.2a over a drop-down menu. In a second step the user has to select a plot type (Figure 4.2b). For ultrafine particles all plots are available, for ozone and carbon monoxide only the *trace plot* and the *heatmap summary plot*. Figures 4.2c, 4.2d and 4.2e show the required parameters for each plot. When the user requests a plot, a new browser window opens with the visualization. For the precomputed maps, the user can switch between the single months by either selecting the desired month in a drop-down menu or by requesting the next or previous month with buttons (see Figure 4.2f). The visualization of the plots is explained in Section 3.2, the screen shots of the three available plots are shown in Figures 3.6, 3.7 and 3.8.

4.5 Efficient Data Handling

It is important to process the requested data fast and efficiently to achieve that the user-interface stays responsive and the user has a smooth experience. The time to compute a certain statistical summary or plot is mostly predetermined by the chosen algorithm. It is possible to optimize the code that the efficiency is increased, but this approach is limited. It is therefore important, to take notice of two other approaches: Precomputing of summaries and efficient handling of raw data. Precomputing is already discussed in Section 4.2. Speeding up the generation time of custom plots can be achieved by optimizing access times to the raw data samples. Table 4.1 provides an overview of different data-formats and their access times for reading two weeks of UFP-data ($1\,636\,716$ samples). The differences are substantial, reading the data from a MySQL table takes over 1300 times as long as when the data is stored in the Matlab format. For the web-application in this thesis, only the Matlab-format and the HDF5-Format are sufficiently fast. For the timing results of the HDF5- and the Matlab file must be considered, that the data for the Matlab file first had to be casted to a common data-type. On one hand data of the same type is loaded faster than mixed data, but on the other hand the data size increases because all values are upcasted to the lowest-common-denominator data type. Both formats are suitable candidates to store the cleaned data (shown in Figure 4.1). For this thesis the HDF5-format is chosen, because it is very efficient for large data-sets and more general than the Matlab format.

Table 4.1: Overview of access-times to read two weeks of UFP raw data (01.01.2014 - 14.01.2014, $1\,636\,716$ samples).

data format	access time [s]
MySQL	85.258
HDF5	0.084
MAT	0.064
CSV	2.778
Pickle	3.706

Chapter 5

Results

5.1 Clean up Process and Calibration

This section explains the clean up process of the raw measurement samples and their calibration. Section 5.1.1 presents the results of the clean up process for the pollutants calibrated with the multi-hop mechanism (ozone and carbon monoxide). Section 5.1.2 evaluates a specific example of the multi-hop calibration for ozone. Section 5.1.3 presents an evaluation of the multi-hop calibration for the whole dataset. Finally, Section 5.1.4 presents the results of cleaning up the UFP data.

5.1.1 Clean up process of Ozone and Carbon Monoxide Measurements

Marking of Possibly Defective Measurements

Finding and marking defective measurements improves the dataset and is useful for making statements about the quality of the data. To achieve this goal, a copy of the data is filtered with a Hampel filter as described in Section 3.1.2 and all samples affected by this filter are marked as possibly defective. Figure 5.1 shows the results of this algorithm for three examples. The first subplot of each figure shows uncalibrated data. The middle subplot depicts the same data but with the Hampel filter applied. In the last subplot all measurements are marked, where a difference between the original data and the filtered data is detected.

Figures 5.1a and 5.1b show that the Hampel filter detects prominent values that are possible outliers. The measurements of the carbon monoxide sensor in Figure 5.1b exhibit more outliers than the measurements of the ozone sensor in Figure 5.1a. This result is expected because the carbon monoxide sensors are known to be more noisy than the ozone sensors.

Figure 5.1c shows that the approach with the Hampel filter is limited. The measured ozone concentration is rising during the morning to a value of around $100\,\mathrm{ppb}$ at twelve o'clock. After this point in time the sensor does not seem to work appropriately. The measured concentration drops to around $10\,\mathrm{ppb}$ and rises again to around $100\,\mathrm{ppb}$ almost instantaneously on multiple occasions. Comparing these measurements to measurements of other days of the same sensor reveals, that the low values of around $10\,\mathrm{ppb}$ in the afternoon of the 23.04.2014 are quite unlikely to be correct and are assumed to be false. The Hampel filter recognizes the erroneous measurement values in the first half of the afternoon up to around three o'clock. After this point in time, the rate of faulty samples increases and the Hampel filter identifies correct samples as faulty and vice versa. This shows that this approach is of limited reliability.

Possible ways to improve for the detection of faulty measurement are more advanced algorithms as mentioned in Section 2.1.3, for example learning based methods as proposed by Sharma et al. [27], where they use training data to generate models for normal and defective sensor measurements which allow to recognize faulty sensor readings.

The Hampel filter marked 1.8% $(239\,570$ out of $12\,843\,994)$ samples as possibly defective for the ozone measurements and for the carbon monoxide measurements 6.4% $(2\,083\,274$ out of

Table 5.1: Calibration path details for graph shown in Figure 5.3 and additional information to diagrams in Figure 5.4. R is the correlation coefficient, R^2 is the coefficient of determination, and α and β are the calibration parameters denoted in Equation 3.1 on page 17.

	devices		checkpoints		calibration parameters		
	parent id	child id	#	R	α	β	R^2
Hop 1	ref-station	7	338	0.785	6.470	1.143	0.61622
Hop 2	7	13	303	0.8524	13.761	0.859	0.72658
Hop 3	13	11	380	0.85651	13.042	0.604	0.7336

$32\,427\,532$) in the cleaned up dataset.

Data Filtering

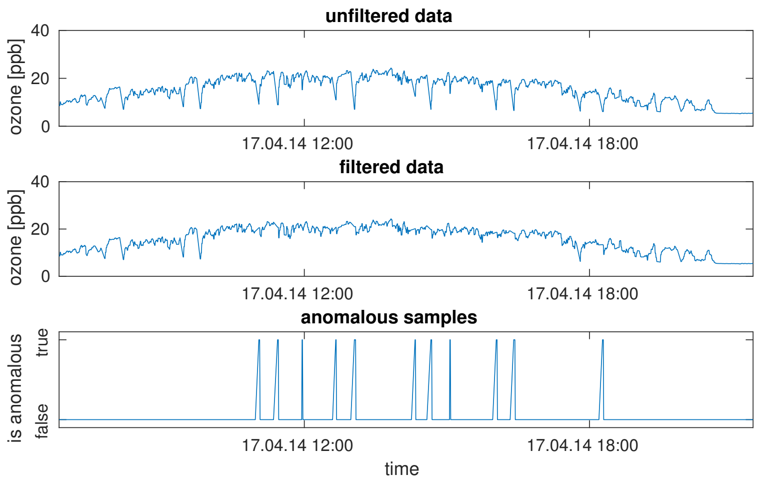
Figure 5.2 visualizes the number of available measurements before and after the clean up process. Figures 5.2a and 5.2c display the number of measurements acquired per week for ozone and carbon monoxide respectively. Figures 5.2b and 5.2d show the number of accumulated samples over time for ozone and carbon monoxide respectively. Before the clean up process, $21\,961\,056$ samples are available for ozone and after that $12\,843\,994$. In total, 58.5% of the originally available samples remain. For carbon monoxide the number drops from $43\,953\,907$ samples to $32\,427\,532$, this means that after the filtering 73.8% of the original samples are still available.

5.1.2 Evaluation of Multi-hop Calibration Example Path

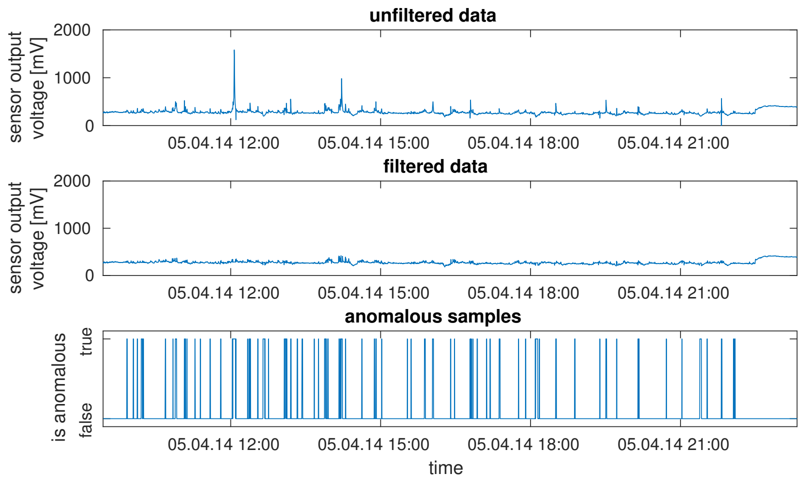
This section shows an example of the multi-hop calibration. The raw data taken for this example is gathered from the ozone sensors measured in May 2014. The threshold for the number of checkpoints is set to 150 and the one for the correlation is set to 0.65. These thresholds differ from the ones introduced in Section 3.1.3, they were adapted to get a path with at least three hops to get a demonstrative example for this section. With the computed checkpoints, the calibration-path extracting algorithm finds six different paths as shown in Figure 5.3. In this section an evaluation of the calibration path marked in red is presented, containing the reference station and the trams 7, 13 and 11 in this order.

The common checkpoints between a calibrated parent and an uncalibrated child are used to perform either ordinary least squares regression for the first hop or geometric mean regression for the subsequent hops. Figures 5.4a, 5.4b and 5.4c provide an overview of the checkpoints between a parent and a child. The top part in each sub-figure presents a scatter-plot of the checkpoints before the calibration was performed, the bottom part presents the scatter-plot after the calibration. Detail about the individual hops are shown in Table 5.1. All calibrations are performed with over 300 checkpoints and the correlation coefficient is at least 0.78 for all of them. The calibration algorithm computes the calibration parameters according to Equation 3.1 on page 17. The resulting intercepts (α) are between 6.470 and 13.042 and the slopes (β) are between 0.604 and 1.143 for this particular path marked with red color in Figure 5.3. The coefficient of determination (R^2) at each hop lies between 0.616 and 0.734.

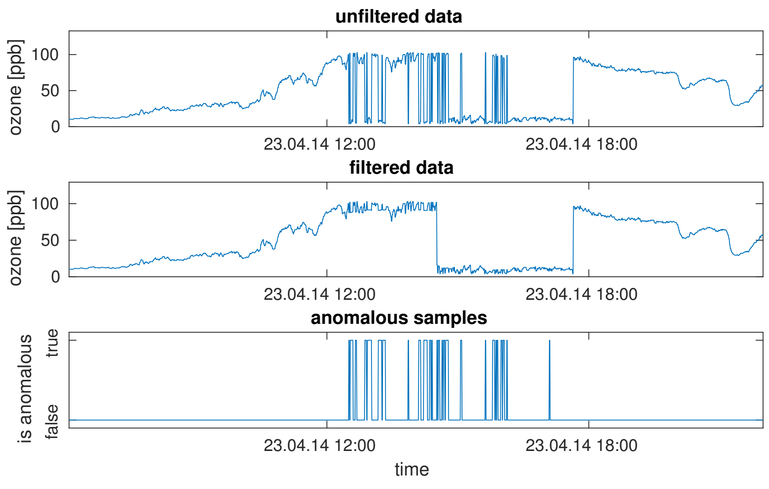
Figure 5.5 shows the absolute error between the reference station and the trams 7 and 11 before and after the calibration. For tram 13 no error is available because this tram has no common checkpoints with the reference station. The mean absolute error in the first hop between the reference station and tram 7 is $9.5057\,\mathrm{ppb}$, after the calibration the mean absolute error is decreased to $5.7525\,\mathrm{ppb}$. The mean absolute error between tram 11 and the reference station is $12.1779\,\mathrm{ppb}$ before the calibration, and $9.5414\,\mathrm{ppb}$ after. The corresponding checkpoints are shown in Figure 5.4a for tram 7 and in Figure 5.4d for tram 11. Tram 11 has only a correlation coefficient of 0.602 for the common checkpoints with the reference station which is too low to be directly calibrated with the reference station.



(a) Uncalibrated measurements of the ozone sensor of tram 4.

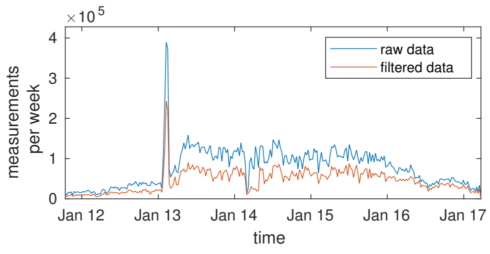


(b) Uncalibrated measurements of the carbon monoxide sensor of tram 5.

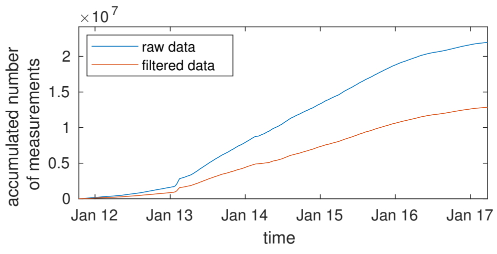


(c) Uncalibrated measurements of the ozone sensor of tram 8.

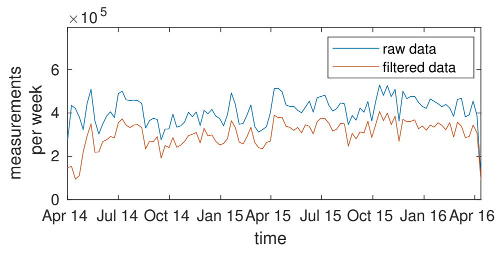
Figure 5.1: Examples of detection of anomalous data samples with a Hampel filter.



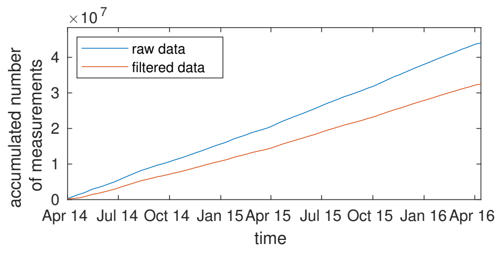
(a) Acquired measurements per week of ozone sensors on trams.



(b) Accumulated number of measurements of ozone sensors on trams.



(c) Acquired measurements per week of carbon monoxide sensors on trams.



(d) Accumulated number of measurements of carbon monoxide sensors on trams.

Figure 5.2: Overview of acquired samples of mobile sensors mounted on trams for ozone and carbon monoxide before and after the filtering process.

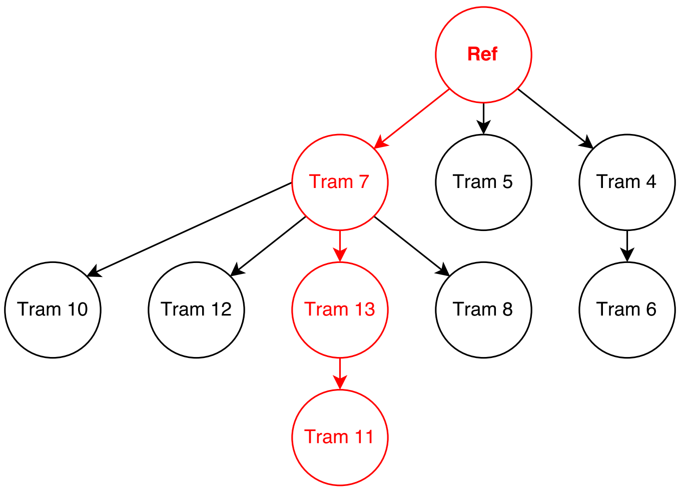
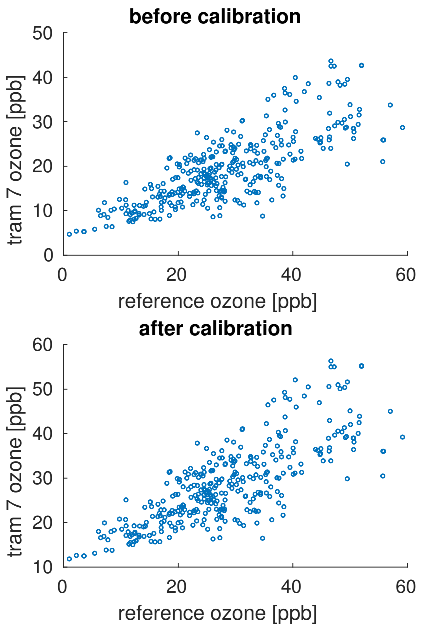
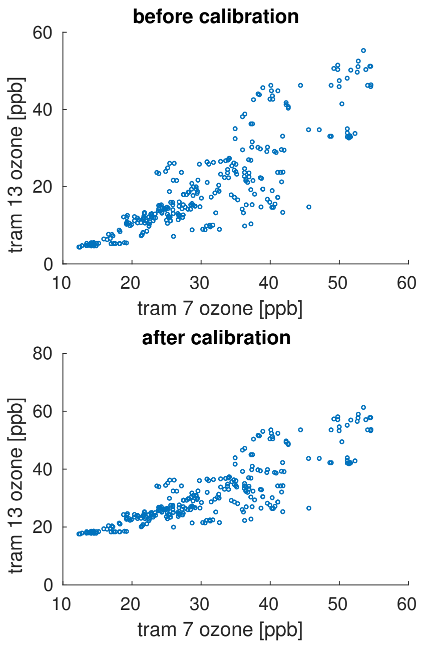


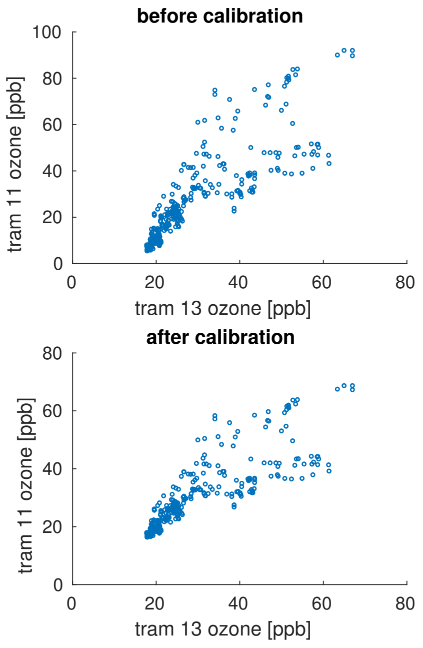
Figure 5.3: Calibration path for ozone data recorded in May 2014.



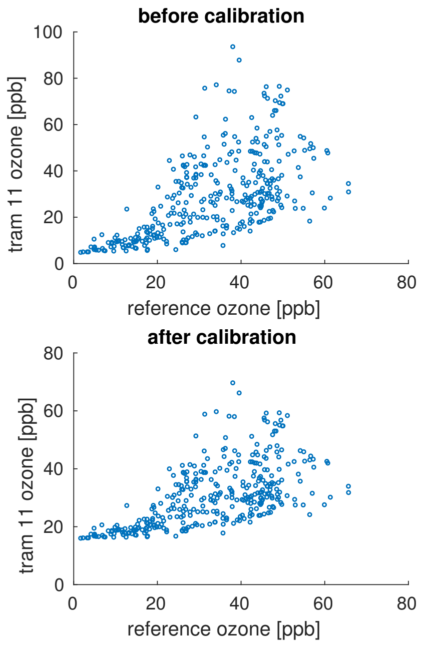
(a) Checkpoints of first hop between reference station and tram 7.



(b) Checkpoints of second hop between tram 7 and tram 13.

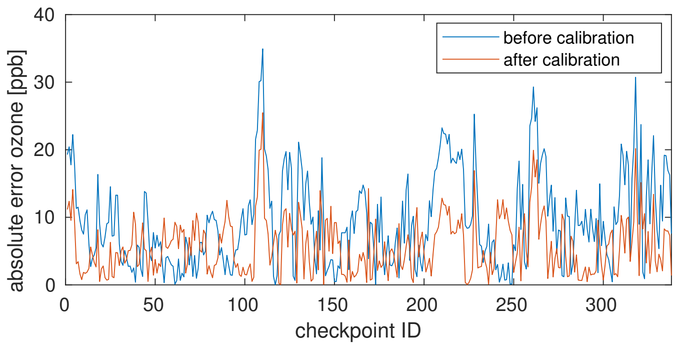


(c) Checkpoints of third hop between tram 13 and tram 11.

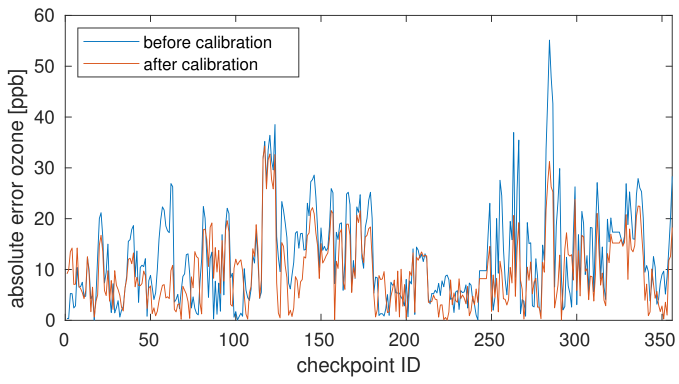


(d) Checkpoints between reference station and tram 11.

Figure 5.4: Scatter plots of ozone measurements before and after performing multi-hop calibration.



(a) Absolute error before (mean: $9.5057\,\mathrm{ppb}$) and after (mean: $5.7525\,\mathrm{ppb}$) calibration in first hop between checkpoints of reference station and tram 7.



(b) Absolute error before (mean: $12.1779\,\mathrm{ppb}$) and after (mean: $9.5414\,\mathrm{ppb}$) calibration in third hop between checkpoints of reference station and tram 11.

Figure 5.5: Absolute error between the checkpoints of the reference station and the trams 7 and 11. For tram 13 no error can be calculated because this tram has no common checkpoints with the reference station.

5.1.3 Evaluation of Multi-hop Calibration for whole Dataset

This section evaluates the multi-hop calibration for the whole dataset. In this thesis only a subset of the whole available dataset is used. The measurements of ozone cover the timespan from 14.10.2011 to 24.03.2017 with a total of $21\,961\,055$ samples. The measurements for carbon monoxide cover the timespan from 01.04.2014 to 13.04.2016 with a total of $43\,953\,906$ samples.

To evaluate the quality of the multi-hop calibration, different metrics are computed. This evaluation focuses on the number of available calibration parameters and on the *normalized root-mean-square deviation* (NRMSD) which is explained later in this section.

Number of Available Calibration Parameters

The quality requirements for checkpoints introduced in Section 3.1.3 are influencing the number of calibration parameters. The larger the thresholds for the number of required checkpoints and the correlation coefficient become, the fewer parameters are available. The reason for this phenomenon is that the demands on the checkpoints of a tram increase and the probability that it can not be inserted in any path increases, and for those trams no calibration parameters can be computed. The numbers of totally available calibration parameters are shown in Table 5.2 for ozone and in Table 5.3 for carbon monoxide. The data of these tables reveals that particularly the correlation threshold has a high influence on the number of available checkpoints, e.g. for carbon monoxide with a fixed threshold of a minimum of 50 checkpoints, the number of available checkpoints drops by 82.6% from 927 to 161. The highest drop for ozone occurs for a fixed threshold of 100 available checkpoints where the value drops by 5.4% from 1117 to 1057. A possible reason for this difference could be that the measurements of carbon monoxide show a higher noise level and have therefore a generally lower correlation.

Table 5.2: Number of available calibration parameters in whole dataset for ozone with different thresholds.

		threshold correlation				
		-1.0	0.3	0.4	0.5	0.6
threshold #checkpoints	30	1171	1143	1133	1128	1120
	50	1168	1137	1129	1124	1116
	100	1117	1099	1092	1073	1057

Table 5.3: Number of available calibration parameters in whole dataset for carbon monoxide with different thresholds.

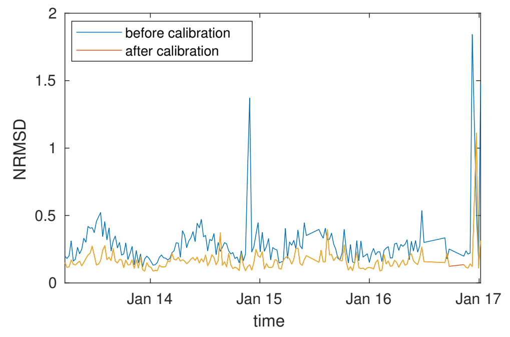
		threshold correlation				
		-1.0	0.3	0.4	0.5	0.6
threshold #checkpoints	30	927	920	866	707	172
	50	927	910	855	666	161
	100	926	897	843	638	161

Error Evaluation

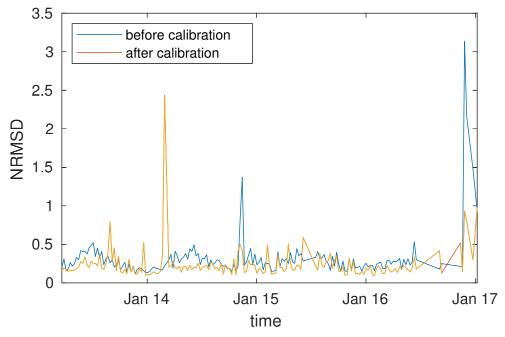
Calibration parameters are computed weekly as described in Section 3.1.3. To evaluate the quality of the calibration for one week, the *normalized root-mean-square deviation* (NRMSD) is used.

Let C be the set of checkpoints in a certain week with $N \in \mathbb{N}$ checkpoints in total. Every checkpoint $C_n \in C$ contains a measurement value y_{ref_n} for the reference station and a measurement value y_{tram_n} for the tram. To evaluate the quality of the calibration for one week, the *normalized root-mean-square deviation* (NRMSD) based on the measurement range of the reference station is used. The NRMSD is defined as

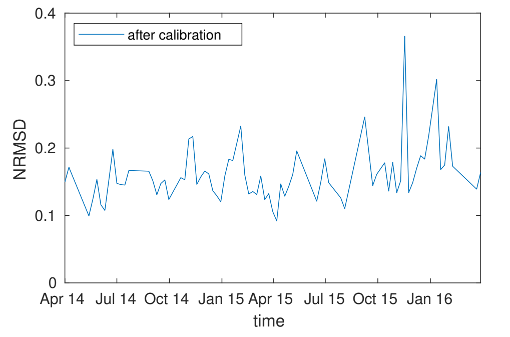
$$NRMSD = \frac{\sqrt{\frac{1}{N} \cdot \sum_{n=1}^{N} (y_{ref_n} - y_{tram_n})^2}}{RANGE},$$



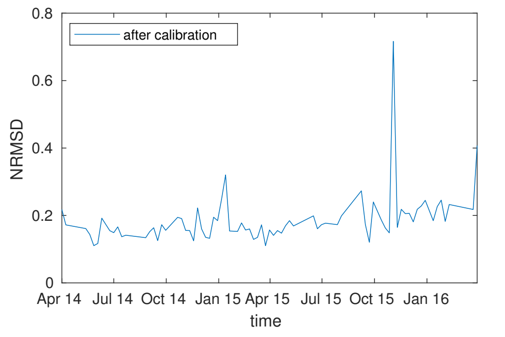
(a) NRMSD of ozone, calibration parameters not shifted. Mean before calibration: 0.293, mean after calibration: 0.165.



(b) NRMSD of ozone, calibration parameters shifted by 2 weeks. Mean before calibration: 0.319, mean after calibration: 0.242.



(c) NRMSD of carbon monoxide, calibration parameters not shifted. Mean after calibration:0.160.



(d) NRMSD of carbon monoxide, calibration parameters shifted by 2 weeks. Mean after calibration:0.186.

Figure 5.6: NRMSD for ozone and carbon monoxide evaluated for the whole dataset with and without a calibration shift of two weeks. The threshold for the minimum number of checkpoints is set to 100 and the one for the minimal correlation coefficient to 0.5.

where

$$RANGE = \max(y_{ref}) - \min(y_{ref}).$$

The NRMSD is computed before and after the calibration for ozone. For carbon monoxide the NRMSD is only computed after the calibration, because the uncalibrated data is measured as an output voltage of the sensor that can not be directly compared to the values of the reference station. The NRMSD for each pollutant is evaluated in two ways, in the first case the data of one week is calibrated with the corresponding calibration parameters of the same week (without time shift). In the second case, the raw data of the same week is calibrated with parameters that are two weeks old (with time shift). The second case is executed because it uses calibration parameters that are outside the calibration window and are therefore not influenced by the raw data to be calibrated.

Evaluation of Calibration with Standard Thresholds In a first part, the quality of the calibration parameters is evaluated for the default thresholds used for the web-tool as described in Section 3.1.3, the threshold for the minimal number of checkpoints is set to 100 and the minimal correlation coefficient is set to 0.5. The evaluation is illustrated in Figure 5.6.

Figure 5.6a shows the NRMSD for ozone before and after the calibration without a time shift of the calibration parameters. Figure 5.6b is the same configuration, but calibration parameters were two weeks old. In the first case the mean of the error could almost be halved from 0.293 to 0.165. In the second case the error is still decreasing from 0.319 to 0.242 but not as far as in the first case. This result is expected because in the first case the calibration parameters are influenced by the data to calibrate which leads to better results.

Figure 5.6c shows the evaluation for carbon monoxide without a shift and Figure 5.6d shows the evaluation with a shift of the calibration window by two weeks. The mean of the NRMSD is in the first case 0.160 and in the second case 0.186.

These results show that the calibration of ozone and carbon monoxide works reliable for the standard thresholds which are a minimum of 100 checkpoints and a minimal correlation coefficient of 0.5 per hop.

Evaluation of Calibration with other Thresholds The previous section *"Number of Available Calibration Parameters"* shows how different threshold values influence the number of totally available calibration parameter. An increasing correlation threshold decreases the number of available calibration parameters, especially for carbon monoxide. This sections evaluate the effect of a lower correlation on the calibration error. Figures 5.7 and 5.8 visualize the NRMSD for ozone and carbon monoxide respectively for a fixed threshold determining the minimal needed number of checkpoints to a value of 100 and changing values for the correlation threshold. In this thesis the correlation thresholds -1, 0.3, 0.4, 0.5 and 0.6 are tested. This evaluation is performed with two weeks old calibration data, such that the used calibration parameters are independent from the checkpoints.

Figure 5.7 shows the plots for ozone. The mean of the NRMSD is always close to 0.32. This value varies slightly, because checkpoints of a tram is removed if it is not part of any calibration path and therefore the set of checkpoints is not the same for each graph. The mean of the NRMSD after the calibration is always between 0.22 and 0.25.

Figure 5.8 shows the plots for carbon monoxide. The mean values of the NRMSD after the calibration are also in a narrow window like the ones for ozone. For carbon monoxide the values vary between 0.17 and 0.20.

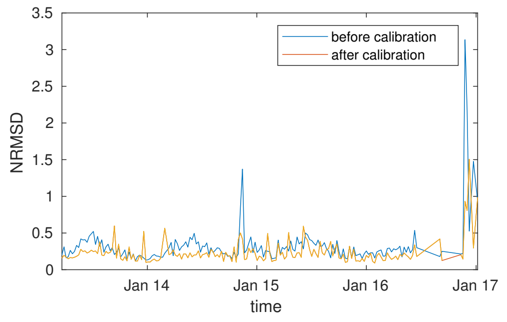
The constantly small errors after the calibration seem odd at first sight. However, one must consider different aspects how the thresholds influence this error:

- This error evaluation is only based on common checkpoints between a reference station and a tram. It gives no insight on how big the error gets at the end of the last hop, if this hop has no common checkpoints with the reference station.
- The lower the correlation threshold is, the shorter the paths get in general. This means, that every tram satisfying the threshold for the minimal number of checkpoints is directly calibrated with a reference station. Like this, the error does not accumulate over multiple hops.
- The bigger the correlation threshold gets, the higher the number of trams with no correlation partner gets whose samples are deleted. This leads to the effect, that checkpoints of the most unreliable tram sensors get removed and do therefore not appear in this error evaluation.

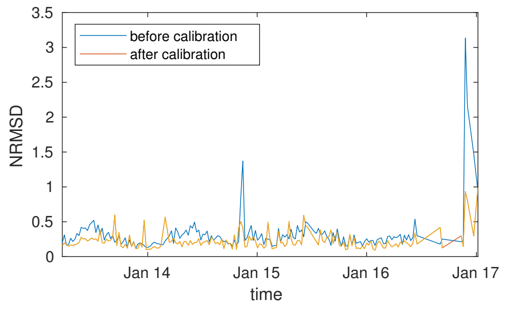
These are reasons why a small correlation threshold can have a positive influence on this error evaluation and vice versa. Therefore this evaluation is not very informative to determine good standard thresholds usable for the toolchain. Due to this reasons, the same thresholds were taken as by Saukh et al. [26] who stated that a correlation threshold of 0.5 delivers reliable results.

5.1.4 Clean up Process and Calibration of Ultrafine Particles Data

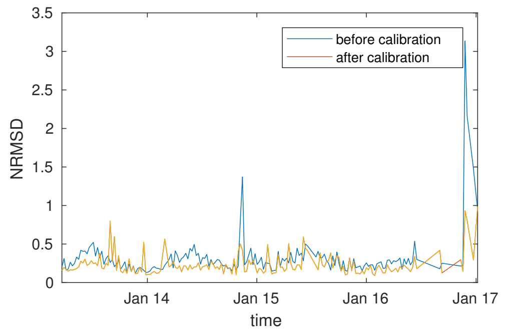
Figure 5.9 gives an overview about the acquired measurement samples for ultrafine particles of the mobile sensors mounted on trams before and after the filtering process. Figure 5.9a presents the acquired measurement per week, Figure 5.9b shows the accumulated samples over time. The filtering- and clean up process is not further discussed in this evaluation, because the original filtering and calibration algorithm is taken from the original project and not altered. Before the clean up process $81\,638\,792$ samples are available, after the calibration $31\,569\,600$. This means that after the clean up process 38.7% of all samples remain.



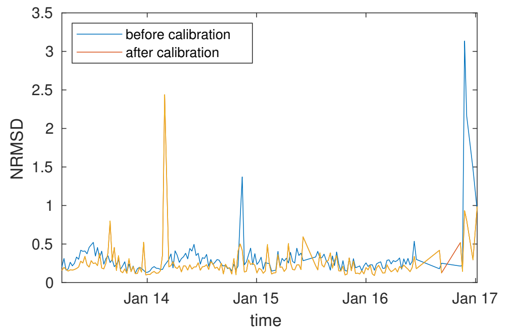
(a) NRMSD for ozone with correlation threshold of -1.0. Mean before calibration: 0.323, mean after calibration: 0.236.



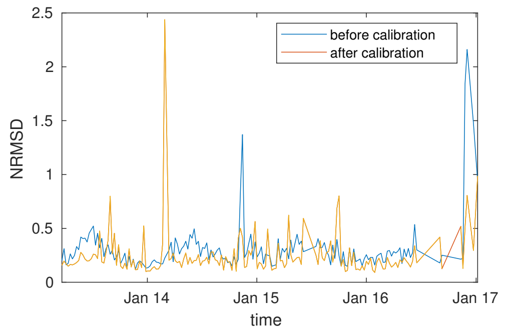
(b) NRMSD for ozone with correlation threshold of 0.3. Mean before calibration: 0.320, mean after calibration: 0.229.



(c) NRMSD for ozone with correlation threshold of 0.4. Mean before calibration: 0.321, mean after calibration: 0.234.

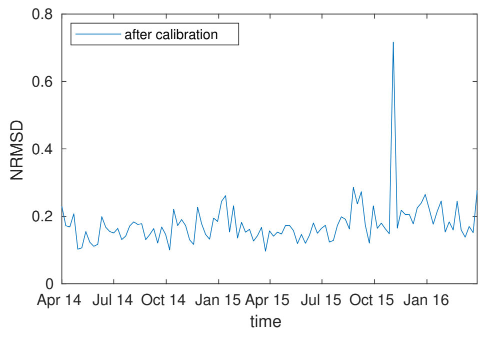


(d) NRMSD for ozone with correlation threshold of 0.5. Mean before calibration: 0.319, mean after calibration: 0.242.

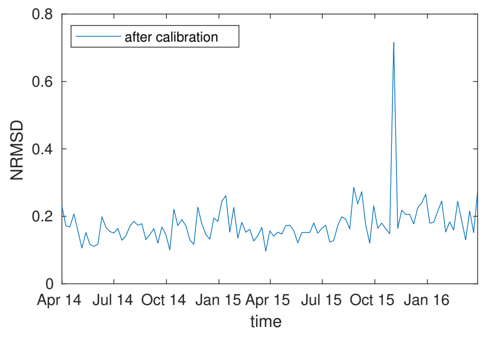


(e) NRMSD for ozone with correlation threshold of 0.6. Mean before calibration: 0.319, mean after calibration: 0.242.

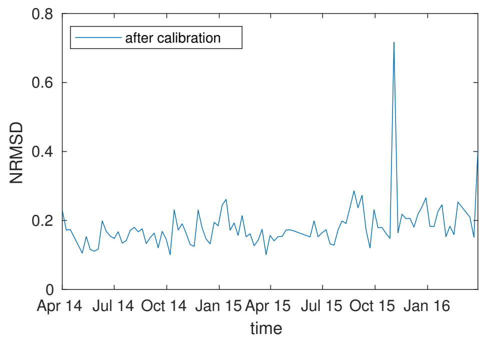
Figure 5.7: NRMSD for ozone for the whole dataset. The data is calibrated with two weeks old calibration parameters. The threshold for the minimum number of checkpoints is set to 100 and the one for the minimal correlation coefficient is different for each subplot.



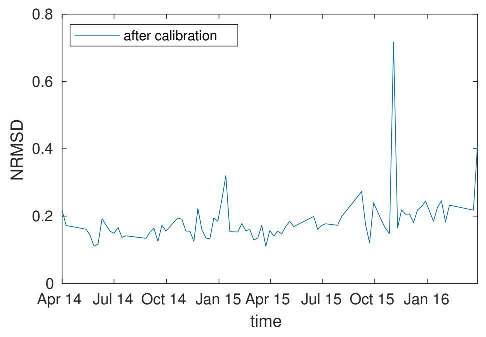
(a) NRMSD for carbon monoxide with correlation threshold of -1.0. Mean after calibration: 0.177.



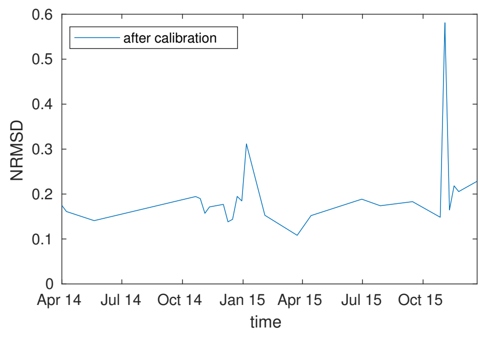
(b) NRMSD for carbon monoxide with correlation threshold of 0.3. Mean after calibration: 0.178.



(c) NRMSD for carbon monoxide with correlation threshold of 0.4. Mean after calibration: 0.182.

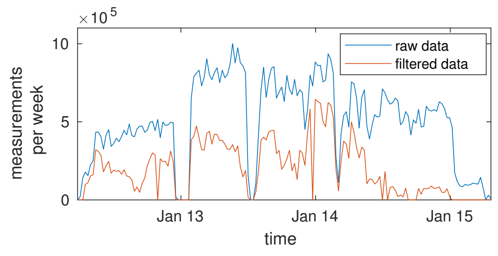


(d) NRMSD for carbon monoxide with correlation threshold of 0.5. Mean after calibration: 0.186.

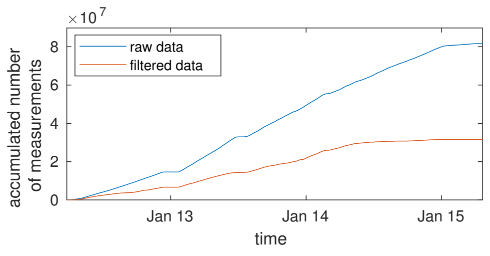


(e) NRMSD for carbon monoxide with correlation threshold of 0.6. Mean after calibration: 0.194.

Figure 5.8: NRMSD for carbon monoxide for the whole dataset. The data is calibrated with two weeks old calibration parameters. The threshold for the minimum number of checkpoints is set to 100 and the one for the minimal correlation coefficient is different for each subplot.



(a) Acquired measurements per week of ultrafine particle sensors.



(b) Accumulated number of measurements of ultrafine particle sensors.

Figure 5.9: Overview of acquired ultrafine particle measurement samples of mobile sensors mounted on trams before and after the filtering process.

5.2 Timing Evaluation of Web-Interface

This section presents the timing evaluation of the plots accessible over the web-interface. The evaluation is performed on a general purpose notebook[1]. The server is running on the same machine as the host, therefore no network delay and rendering time of the web-browser is included in these results. This section presents first the timing evaluation for the *trace plots* in Section 5.2.1. The evaluation for the *heatmap summary plots* is presented in Section 5.2.2 and the one for *ultrafine particles map plots* in Section 5.2.3.

5.2.1 Trace Plots

To test the performance of the *trace plot* generation, different tram- and track-IDs were requested for data of four weeks without applying a day-time filter. To successfully generate a plot, between $0.3\,\mathrm{s}$ and $1.1\,\mathrm{s}$ are needed. However, testing this plot excessively is not possible. For most input configurations consisting of a timespan, a tram ID and a tram track ID the generating tool is not able to extract a single valid trace. The reasons for this problem are:

- The thresholds to determine whether a tram is on a tram track or not are chosen to restrictive and e.g. small errors in the GPS value lead to a discarding of theoretically valid traces.
- Trams change their route and continue to drive on another tram track, or turn before the final halt.
- To many samples are removed in the filtering process, leading to the problem that big gaps occur between consecutive measurements and the algorithm is no longer able to extract the trace correctly.

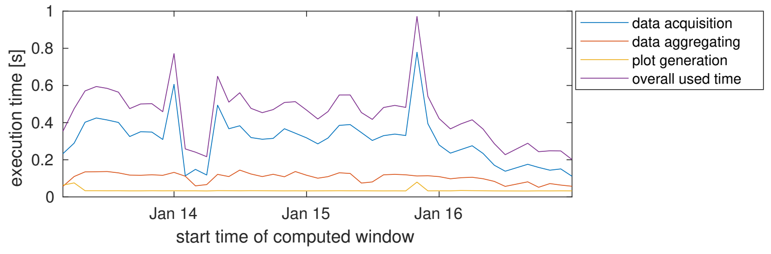
5.2.2 Heatmap Summary Plots

To evaluate the time needed to generate *heatmap summary plots*, the timing to precompute monthly plots for each pollutant is evaluated. This is a reasonable approach, because the preprocessing module generates the same HTML output as the post-processing module. The only difference is, that the file is stored on the hard-disk and not sent to the client over the network. But since network delays are anyway excluded in this evaluation this does not matter. Figure 5.10 shows the timing results. The timing is split in three different parts which are shown in these plots: the time to read the data from the HDF5-files (data acquisition), the time to aggregate the data to cells of size $200\,\mathrm{m} \times 200\,\mathrm{m}$ (data aggregating) and the time to plot the data and store the HTML-file on the hard-disk (plot generation). Additionally, the overall used time is displayed. The mean overall execution time of all pollutants combined is $1.0530\,\mathrm{s}$. The biggest part of this time is caused by the data acquisition with an overall mean of $0.7690\,\mathrm{s}$. The mean for the single pollutants is correlating with the total available samples of each pollutant.

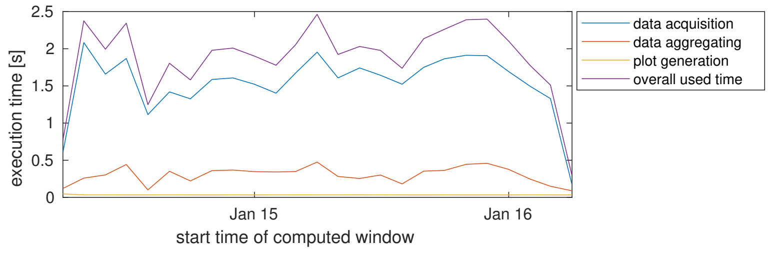
[1]HP EliteBook 8460p with $3.8\,\mathrm{GiB}$ memory, Intel Core i7-2620M CPU @ $2.70\,\mathrm{GHz} \times 4$, 64-bit architecture, running with Ubuntu 16.04 LTS

5.2.3 Ultrafine Particles Map Plots

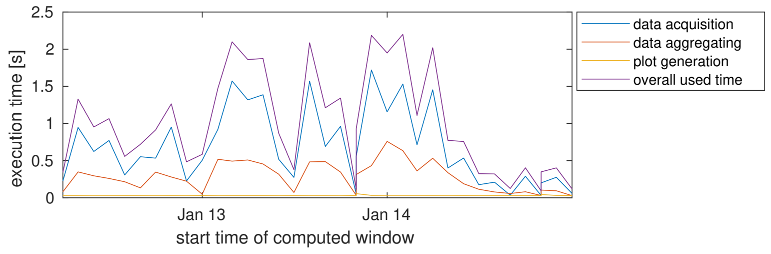
To evaluate the needed time to generate map plots for ultrafine particles, the same evaluation was performed as described in Section 5.2.2, by precomputing monthly maps for all pollutant characteristics of ultrafine particulate matter (diameter, lung-deposited surface area and concentration). Figure 5.11 visualizes the evaluation. The timing is split into loading the calibrated data (data acquisition), aggregating the data to $100\,\mathrm{m} \times 100\,\mathrm{m}$ cells (data aggregation), adding land-use data to the measurements (add statistics), computing the land-use regression model (compute model), generating and storing the plot on disk (generate plot) and the overall used time. This plot needs much time to be generated, the mean overall used time is $16.9797\,\mathrm{s}$, the biggest part of this time is used by the data aggregation to the cells with $14.9161\,\mathrm{s}$.



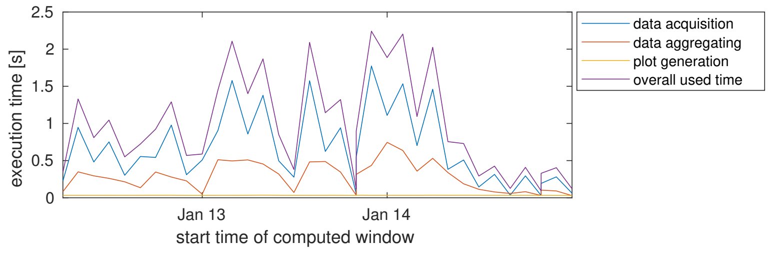
(a) Plot generation time for ozone.



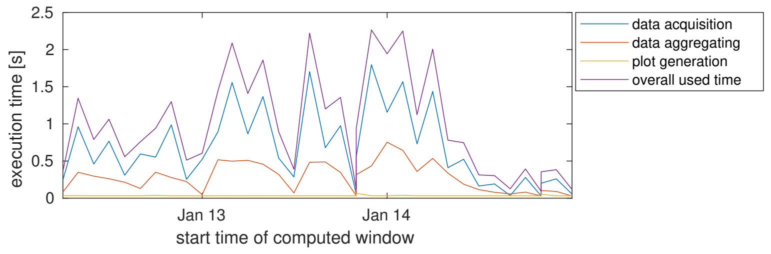
(b) Plot generation time for carbon monoxide.



(c) Plot generation time for ultrafine particles (diameter).

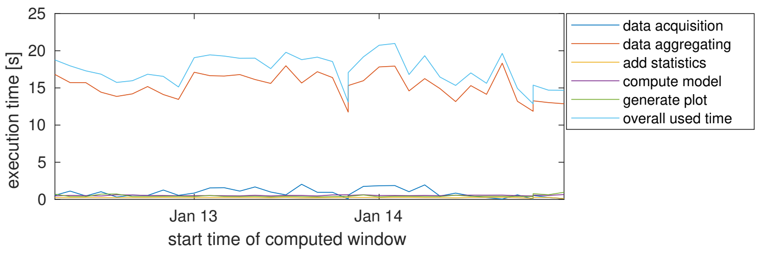


(d) Plot generation time for ultrafine particles (lung-deposited surface area).

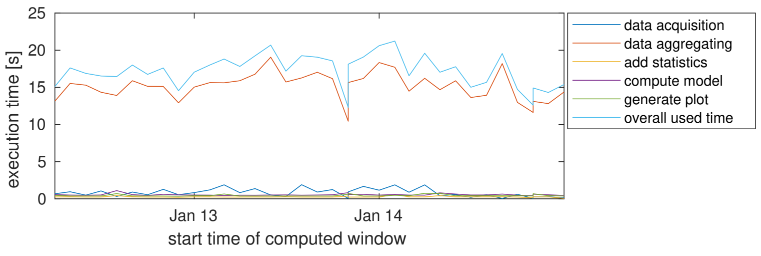


(e) Plot generation time for ultrafine particles (concentration).

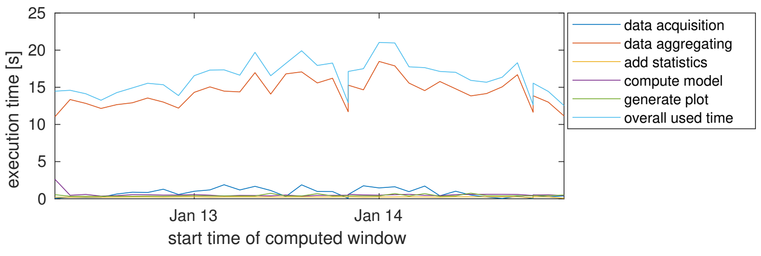
Figure 5.10: Overview of generation times of *heatmap summary plots* generated monthly for all pollutants over the whole available dataset.



(a) Plot generation time for ultrafine particles (diameter).



(b) Plot generation time for ultrafine particles (lung-deposited surface area).



(c) Plot generation time for ultrafine particles (concentration).

Figure 5.11: Overview of generation times of *ultrafine particles map plots* generated on a monthly basis for the whole dataset.

Chapter 6

Summary and Outlook

6.1 Summary

This thesis presents an automated toolchain that is able to filter and calibrate big amounts of air quality data delivered by a mobile sensor network consisting of low-cost sensors mounted on trams that are driving through the city of Zurich. This mobile sensor network is part of the *OpenSense* project headed at ETH Zurich where studies were performed on how to improve the quality of the huge dataset. This thesis focused on the pollutants ozone, carbon monoxide and ultrafine particulate matter. To remove faulty and unusable measurement samples of low quality, various filters are applied. Additionally possibly defective measurements are marked. The measured samples of ozone and carbon monoxide are calibrated with a multi-hop mechanism. In order to calibrate the data, checkpoints between single sensors of trams and reference stations delivering ground-truth are computed. These checkpoints are the base to build calibration paths for multiple devices, where already calibrated sensors are used to calibrate other devices. The measurements of ultrafine particulate matter are calibrated by applying parameters of the sensors, such as the determined null-offset or status variables providing information about the delivered quality of the devices.

The toolchain presented in this thesis is further able to visualize the cleaned up dataset over a web-interface in fast and intuitive ways. Three different plot types are available: A *trace plot* displaying time-series plots of sensors on specific tram tracks, a *heatmap summary plot* visualizing statistical summaries for the measurements on a map, and a visualization of modeled ultrafine particulate matter for the whole city of Zurich generated with a Generalized Additive Model based on land-use parameters.

This thesis provides also a collection of environmental data. Weather data can be used as meta-data to classify measurements collected under similar conditions. Land-use data like building-area density, elevation or traffic-density are helpful to generate models of the pollution load at places where no actual sensors are available.

6.2 Outlook

A first improvement for the existing toolchain could be to precompute time-series of measurements on tram-tracks to decouple the sample extraction from the visualization. A problem with the *trace plot* in its current state is that upon a user request often no measurements are available in the searched time-window. If these traces were precomputed, the user-interace could directly make suggestions to the user on where displayable data is available. Another interesting extension would be to cache frequently requested measurement samples instead of loading them every time from the hard drive. Another improvement would be to add the new land-use data to the existing collection and evaluate influences of these new parameters on the model outcome.

Appendix A

Appendix

A.1 Toolchain Manual

A.1.1 Clean up of Raw Data and Precomputation of Plots

1. Copy whole folder `webtool_complete_all_pollutants` to target device.
2. Create new MySQL database for cleaned-up measurements.
   - Open `SQL/build_cleaned_up_db.sql`.
   - Adapt username and password to target database.
   - Execute file.
3. Adapt database configurations for ozone and carbon monoxide.
   - Open `config_files/database_configuration.py`.
   - Set username and password for raw data DB for `original_data_general`.
   - Set username and passwort for cleaned-up database (`cleaned_up_data_general` and `cleaned_up_data_general_with_db`.
4. Repeat step 3 for ultrafine particles in file `config_files_ufp/database_configuration.py`. Use the same credentials, if ultrafine particles measurements are in the same database as ozone and carbon monoxide measurements.
5. Adapt MySQL queries in `config_files/database_queries.py` and `config_files_ufp/database_queries.py` if the raw measurements are not in the database *opensense_pbl*.
6. Adapt configuration constants (if needed) for filtering and calibrating the data in `config_files/constants.py` for ozone and carbon monoxide and in `config_files_ufp/constants.py` for ultrafine particles.
7. Set the standard calibration thresholds for the multi-hop pollutants in the file `config_files_hdf5/constants.py`. Use only values that exist also in the file `config_files/constants.py` (in dictionary with name `CALIBRATION_PARAMETERS_GENERATION_CONFIG`. The standard calibration thresholds can be changed after the calibration if necessary.
8. Set desired window of ozone and carbon monoxide data to be calibrated of the available raw dataset in file `a000_main_init_multihop.py` for each pollutant (only the variables *start_time* and *end_time*).
9. Set desired window of ufp data to be calibrated in the file `b000_main_init_ufp.py` (only the variables *start_time* and *end_time*).
10. Execute files `a000_main_init_multihop.py` and `b000_main_init_ufp.py`.

A.1.2 Starting Server

1. Open console, execute the command
   ```
   $ FLASK_APP=server.py flask run
   ```
2. Open URL displayed in console

A.1.3 Server Data

- All data needed by the web-server is stored in the folder `serverdata_public` in the toolchain folder (`SVN/code/webtool_complete_all_pollutants`).
- The calibration evaluation for the pollutants calibrated with the multi-hop algorithm are stored in the folder `SVN/code/webtool_complete_all_pollutants/data_handler_out`.

A.1.4 Recommendations before Server Launch

- Sanitize input of received server-requests.
- Setup *Cron job* that periodically empties the content of the folder `SVN/code/webtool_complete_all_pollutants/serverdata_public/temp_plots`, custom plots requested by a user are stored here and do not have to be stored after successful transmission.
- The current implementation does not automatically include new data, if the database containing uncleaned data is updated. To implement this:
  - Multi-hop pollutants:
    * Implement state-retention mechanism that stores the last execution times of the scripts listet in `a000_main_init_multihop.py` (a1, a2, a3, a6, ab1).
    * Whenever 7 days (constant `MYSQL_PARTITIONING_TIMESPAN_DAYS` in `webtool_complete_all_pollutants/config_files/constants.py`) since the last execution of a1 have passed, execute `a1_cleanup_data.cleanup_data_partitioned` and `a2_compute_checkpoints.compute_checkpoints_partitioned`.
    * Execute `a3_generate_calibration_parameters.generate_calibration_parameters` when the previous point was executed. Set the start- and end-time to the previous week, because to compute the calibration parameters, checkpoints of one week before and after the time-window are considered. If the start- and end-time in the previous bullet-point were t_{start} and t_{end}, set them for this script to $t_{start} - 7\text{days}$ and $t_{end} - 7\text{days}$ respectively.
    * append the new cleaned up samples and the calibration parameters to the hdf5 files:
      ```
      # append to hdf5 file
      df = data_handler.get_cleaned_up_raw_data_tram_DB(
      current_start, current_end, selected_pollutant)
      data_manager_cleaned_up_samples.append_block(
      selected_pollutant, current_start, current_end, df)

      # overwrite existing hdf5 file
      df = data_handler.get_calibration_parameters_DB(
      selected_pollutant, 0, 3000000000000)
      data_manager_calibration_parameters.store_calibration_parameters(
      selected_pollutant, df_calibration_parameters)
      ```
  - Ultrafine particles data:

    * Execute `b1_main_populate_cleaned_up_db.generate_and_store_calibrated_data` whenever a week past since the last execution.
    * Append new block to hdf5 files for each different UFP type (concentration, LDSA, diameter) as described in `b2_ufp_populate_hdf5_with_cleaned_up_data.populate_hdf5` but without reseting the files first.

A.2 Paths to Code

- Automated tool-chain: `SVN/code/webtool_complete_all_pollutants`
- Environmental datasets:
  - GPS tracks of trams: `SVN/code/generate_tram_gps_path`
  - Weather data acquisition: `SVN/code/data_acquisition_weather_pollution`
  - Landuse dataset for building density: `SVN/code/landuse_buildings`
  - Landuse dataset for elevation: `/SVN/code/landuse_elevation`
  - Landuse dataset for traffic: `SVN/code/landuse_roads`

A.3 Required Software and Installation Steps to run the Toolchain

These steps were executed on Ubuntu 14.02 LTS

1. Install mysql-server:
   ```
   $ sudo apt-get install mysql-server
   ```
2. Install Anaconda:
   - Go to `https://docs.anaconda.com/anaconda/install/linux`
   - Download installer for Python 2.7 (Anaconda 5.0.0 For Linux Installer)
   - Execute downloaded installer:
     ```
     $ bash ~/Downloads/Anaconda2-5.0.0-Linux-x86_64.sh
     ```
3. Install additionally needed Anaconda packages:
   ```
   $ conda install mysql-connector-python
   $ conda install mysql-python
   $ conda install plotly
   $ conda install rpy2
   $ conda install shapely
   $ conda install -c conda-forge r-r.utils
   $ conda install rpy2
   ```
   (rpy2 is listed two times, because *conda-forge r-r.utils* downgraded certain required packages)

   A more detailed list with console output of the installation steps can be found in the file `SVN/documentation/install_log`, a list of all installed Conda packages is provided in the file `VN/documentation/list_installed_conda_packages`.

Appendix B

Project Schedule

Week #	Tasks
Week 5	Building/Traffic model(s)
Week 6	Creating cleaned up Database
Week 7	Creating cleaned up Database
Week 8	Creating cleaned up Database
Week 9	(Backup)
Week 10	“Big” model implementation (Traffic, season, weather, industry emmissions, OpenSense measurements, . . .)
Week 11	Air quality index
Week 12	Post-processing module: map generation, statistical summaries
Week 13	Post-processing module: map generation, statistical summaries
Week 14	Post-processing module: map generation, statistical summaries
Week 15	Preprocessing module + Database setup
Week 16	Preprocessing module + Database setup
Week 17	Browser Frontend
Week 18	Browser Frontend / Backend: Visualization Module
Week 19	Backend: Visualization Module
Week 20	Augmented Reality Application
Week 21	Augmented Reality Application
Week 22	Augmented Reality Application
Week 23	(Backup)
Week 25	Report
Week 26	Report

Appendix C

Declaration of Originality

Appendix D

Task Description

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Prof. L. Thiele

Semester Thesis / Master Thesis / Group Work:

Visualize Volumes of Air Quality Data

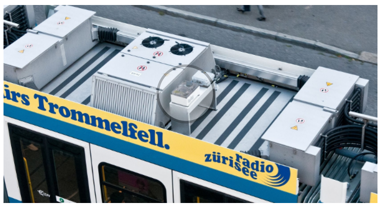
Motivation: Urban air pollution is a major concern in modern cities. Atmospheric pollutants are responsible for a variety of respiratory illnesses (*e.g.*, asthma) and some are known to cause cancer if humans are exposed to them for extended periods of time. Traditionally, air pollution is monitored by a network of static measurement stations operated by official authorities. The extensive cost of acquiring and operating static stations severely limits the number of installations.

In recent years new gas sensors appeared on the market, which are inexpensive, small, and suitable for mobile measurements (see Fig. 1(a)). As part of the OpenSense project, we integrated several such low-cost sensors into compact air pollution monitoring stations, which are deployed on top of several trams in Zurich, as shown in Fig. 1(b). We daily gather megabytes of air quality data and store these in our database. Currently, we use only basic time-series plots and maps to visualize the data. These give little insight into the data and, thus, user experience with the web site can be significantly improved.

Task: The goal of this thesis is to design and implement a set of data presentation and visualization tools for our air quality data set. Your tools should include data summaries and statistical data properties of the real-time and historical data to improve user experience. You are welcome to suggest and implement your own visualization and data analysis ideas, but the web page must be robust.

This involves for you the following tasks:

(a) Low-cost gas sensors.



(b) Tram with a sensor node on top.

Figure 1: Air pollution monitoring on top of trams in Zurich.

- Design a set of *data summaries* that show interesting aspects of gathered data and suggest how to efficiently implement them for a big data set.
- Write software tools that *compute statistics and visualize the air quality data* for various time scales and locations, make use of *pre-computation and caching*.
- Integrate your tools in a *web-page prototype* and show a demo at your final presentation.
- (Master Thesis only!) Integrate your tools into an *augmented or virtual reality application*, *e.g.* for Google cardboard. We are of course also open for your own ideas!

OpenSense web page: www.opensense.ethz.ch

Requirements: Interest in data analysis and visualization of big data sets.

Interested? Please have a look at http://www.tec.ethz.ch/research.html **and contact us for more details!**

Contacts

- Balz Maag: bmaag@tik.ee.ethz.ch, ETZ G75
- Zimu Zhou: zzhou@tik.ee.ethz.ch, ETZ G85

Bibliography

[1] Air quality monitoring stations in zurich. https://awel.zh.ch/internet/baudirektion/awel/de/luft_klima_elektrosmog/luftqualitaet.html. Accessed: 2017-09-06.

[2] Auswirkungen der Luftverschmutzung auf die Gesundheit. https://www.bafu.admin.ch/bafu/de/home/themen/luft/fachinformationen/auswirkungen-der-luftverschmutzung/auswirkungen-der-luftverschmutzung-auf-die-gesundheit.html. Accessed: 2017-09-06.

[3] Elevation data documentation. https://dds.cr.usgs.gov/srtm/version2_1/Documentation/Quickstart.pdf. Accessed: 2017-09-19.

[4] Elevation data provided by srtm. https://dds.cr.usgs.gov/srtm/version2_1/SRTM3/Eurasia/. Accessed: 2017-04-11.

[5] Google maps website. https://www.google.ch/maps/. Accessed: 2017-06-19.

[6] Hampel filter. https://ch.mathworks.com/help/signal/ref/hampel.html. Accessed: 2017-09-22.

[7] Leaflet website. http://leafletjs.com/. Accessed: 2017-09-18.

[8] Opensense web-interface. http://data.opensense.ethz.ch/.

[9] Opensense webpage. http://www.opensense.ethz.ch.

[10] Openstreetmap buildings. http://wiki.openstreetmap.org/wiki/Buildings. Information, how buildings are mapped in dataset. Accessed: 2017-09-19.

[11] Openstreetmap website. http://www.openstreetmap.org. Accessed: 2017-09-18.

[12] Plotly scattermapbox documentation. https://plot.ly/python/scattermapbox/. Accessed: 2017-09-18.

[13] Plotly website. https://plot.ly/python/. Accessed: 2017-09-18.

[14] Srtm topography. https://dds.cr.usgs.gov/srtm/version2_1/Documentation/SRTM_Topo.pdf. Accessed: 2017-09-19.

[15] Traffic count system city of zurich. http://www.verkehrsmanagement.ch/. Accessed: 2017-03-31.

[16] Viewranger website. http://www.viewranger.com. Accessed: 2017-06-19.

[17] Eiman Elnahrawy and Badri Nath. Cleaning and querying noisy sensors. In *Proceedings of the 2Nd ACM International Conference on Wireless Sensor Networks and Applications*, WSNA '03, pages 78–87, New York, NY, USA, 2003. ACM.

[18] D. Hasenfratz, O. Saukh, C. Walser, C. Hueglin, M. Fierz, and L. Thiele. Pushing the spatio-temporal resolution limit of urban air pollution maps. In *Proc. of PerCom*, pages 69–77. IEEE, 2014.

[19] M. Keller, J. Beutel, O. Saukh, and L. Thiele. Visualizing large sensor network data sets in space and time with vizzly. In *37th Annual IEEE Conference on Local Computer Networks - Workshops*, pages 925–933, Oct 2012.

[20] Jason Jingshi Li, Boi Faltings, Olga Saukh, David Hasenfratz, and Jan Beutel. Sensing the air we breathe — the opensense zurich dataset, 2012.

[21] Balz Maag, Zimu Zhou, Olga Saukh, and Lothar Thiele. Scan: Multi-hop calibration for mobile sensor arrays. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(2):19:1–19:21, June 2017.

[22] M Mourad and Jean-Luc Bertrand-Krajewski. A method for automatic validation of long time series of data in urban hydrology. 45:263–270, 02 2002.

[23] Michael D. Mueller, David Hasenfratz, Olga Saukh, Martin Fierz, and Christoph Hueglin. Statistical modelling of particle number concentration in zurich at high spatio-temporal resolution utilizing data from a mobile sensor network. *Atmospheric Environment*, 126:171 – 181, 2016-02. Published online 2 December 2015.

[24] Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and Mani Srivastava. Sensor network data fault types. *ACM Trans. Sen. Netw.*, 5(3):25:1–25:29, June 2009.

[25] N. Ramanathan, L. Balzano, M. Burt, D. Estrin, T. Harmon, C. Harvey, and et al. Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks. 2006.

[26] Olga Saukh, David Hasenfratz, and Lothar Thiele. Reducing multi-hop calibration errors in large-scale mobile sensor networks. In *Proc. of IPSN*, pages 274–285. ACM/IEEE, 2015.

[27] Abhishek B. Sharma, Leana Golubchik, and Ramesh Govindan. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Trans. Sen. Netw.*, 6(3):23:1–23:39, June 2010.

[28] Bo Sheng, Qun Li, Weizhen Mao, and Wen Jin. Outlier detection in sensor networks. In *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '07, pages 219–228, New York, NY, USA, 2007. ACM.

[29] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, and Wei Hong. A macroscope in the redwoods. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys '05, pages 51–63, New York, NY, USA, 2005. ACM.