Faculty of Engineering, Computer Science and Psychology
Institute of Databases and Information Systems

Evaluating Sensor Data in the Context of Mobile Crowdsensing

Master Thesis at Ulm University

Submitted by:
Maximilian Blasi
maximilian.blasi@uni-ulm.de
904922

Reviewers:
Prof. Dr. Manfred Reichert
Prof. Dr. Rüdiger Pryss

Supervisor:
Robin Kraft

2022

Version October 24, 2022

Composition: PDF-LaTeX 2_ε

Acknowledgement

I want to thank Prof. Dr. Manfred Reichert and Prof. Dr. Rüdiger Pryss for reviewing my work. My sincere thanks goes to Daniel Zöllner and Jan-Phillip Stöhr, as well as the rest of my friends and family, for the help and support they provided.

Furthermore, I want to thank my supervisor Robin Kraft for his advice, guidance and the possibility to work on this topic.

Abstract

With the recent rise of the *Internet of Things* the prevalence of mobile sensors in our daily life experienced a huge surge. *Mobile crowdsensing* (MCS) is a new emerging paradigm that realizes the utility and ubiquity of smartphones and more precisely their incorporated smart sensors. By using the mobile phones and data of ordinary citizens, many problems have to be solved when designing an MCS-application. What data is needed in order to obtain the wanted results? Should the calculations be executed locally or on a server? How can the quality of data be improved? How can the data best be evaluated? These problems are addressed by the design of a streamlined approach of how to create an MCS-application while having all these problems in mind. In order to design this approach, an exhaustive literature research on existing MCS-applications was done and to validate this approach a new application was designed with its help. The procedure of designing and implementing this application went smoothly and thus shows the applicability of the approach.

Contents

1 Introduction

With the recent rise of the *Internet of Things* the prevalence of mobile sensors in our daily life experienced a huge surge. Inspired by this trend *mobile crowdsensing* (MCS) [39] is proposed in order to realize a cheaper and more flexible data acquisition and analysis than with traditional sensor networks [69]. Guo *et al.* [46] define MCS as follows: "A new sensing paradigm that empowers ordinary citizens to contribute data sensed or generated from their mobile devices and aggregates and fuses the data in the cloud for crowd intelligence extraction and human-centric service delivery." In other words MCS tackles specific problems by utilizing large groups of individuals. These individuals each have a mobile device with the problem specific sensors (*e.g.*, ***GPS***, ***accelerometer***, ***gyroscope***, and ***magnetometer***) and are tasked with executing some simple sensing tasks. The different habits of the participating individuals cause heterogeneity in the data and the "mobile"-aspect of the participants requires the data coverage and the data quality to be controlled.

An example for the application of MCS is traffic monitoring. The intuitive way of monitoring the traffic would be the installation of traffic monitoring cameras or something similar. But this approach would require a lot of money investment in these cameras. With the help of MCS this task can be done way cheaper and easier, all that is needed are participants that are willing to share their GPS (Global Positioning System) data while driving in a car. If a lot of participants in close proximity are only advancing slowly, the possibility of a traffic jam in this region is high.

Designing an MCS-application is no easy task and comes with many difficulties. The first problem to consider is what sensor data is necessary to achieve the wanted result. As the participants of such an application are in most cases volunteers sensing with their own mobile phone, it is advisable to keep the used sensors to a minimum. More sensors used mean a higher resource cost for the participant and a higher variety of data which can possibly lead to concerns regarding the data privacy of the participants.

The next problem stems from the question "Should the calculations be executed locally or after an upload on the server?", the choice of the processing device. This problem can be boiled down to pretty similar considerations as the previous one. More local calculations lead to a higher resource cost for the participant, but more server-side calculations expose more data privacy issues as the raw data is uploaded and at the same time it also increases the network requirements. These conflicting points lead to no clear solution and instead require a consideration between the trade-offs each time. The next problem concerns the nature of crowdsensed data itself. MCS provides the opportunity to sense data at all times and at all places which leads to huge amounts of data. But all the sensing occurs in completely unsupervised environments which means that a sizeable amount of this data is probably due to unforeseen circumstances faulty or even missing. There is a multitude of possible approaches to this problem but the decision how to best solve it in a specific case can be highly complex. After these fundamental decisions have been made the last problem is finally how to evaluate the data in order to get the wanted results. This problem is extremely application specific which is why there is no universal answer to this problem.

These problems can be summarized in the following problem statements:

- What data is needed in order to obtain the wanted results?
- Where should the calculations be executed?
- How can the quality of data be improved?
- How can the data best be evaluated?

In order to tackle these questions an exhaustive literature research in the form of a systematic review is made in Chapter 2. In Chapter 3, the knowledge gained from this research is used to create a problem approach in the form of a step-by-step guide for how to develop an MCS-application. To validate this approach it is then applied in Chapter 4. Finally, Chapter 5 concludes this work.

2 Handling Sensor Data in the Context of Mobile Crowdsensing

This chapter focuses on already existing literature on the topic of mobile crowdsensing. First it is explained how the research is conducted and what criteria papers need to fulfill in order to be valid for this work. Then the papers are exhaustively studied and all relevant information are extracted. And after that some advanced data management techniques for MCS are presented.

2.1 Methodology

The sources for papers describing MCS-applications were selected in the manner of a systematic review [118], based on the PRISMA Statement [85]. The evaluation of these papers can be seen in Section 2.2.

The main research questions are:

- What are goals for MCS-applications?
- Which sensors are MCS-applications utilizing and how?
- What time constraints do MCS-applications have?
- On which processing device are MCS-applications performing their calculations?
- How are MCS-applications evaluated?
- How can the findings of the paper be best presented?

In order to fulfill these research questions the following inclusion criteria (IC) are defined.

- **IC1:** Any paper explaining its own MCS-application.
- **IC2:** Any paper explaining systems using one or more mobile devices as sensors.

Furthermore the following exclusion criteria (EC) are used to further enhance the applicability and quality of the chosen papers:

- **EC1:** Systems not using any mobile sensors.
- **EC2:** Systems not describing how the data is sensed.
- **EC3:** Any paper older than 2007.
- **EC4:** Articles without full (English) text availability.
- **EC5:** Evaluation methods need to be applicable to MCS-sensed data.
- **EC6:** Any not peer reviewed paper.

In order to get papers fulfilling all these criteria the following search strategy is used on multiple scientific databases (ACM Digital Library [1], IEEE Xplore [51], PubMed [96]):

$$Abstract : (crowdsens^{*}) AND (AllField : (application) OR AllField : (app)) \tag{2.1}$$

A word beginning with *crowdsens* has to be in the abstract or title and anywhere in the paper the word *application* or *app* needs to occur, additionally a filter is applied showing only papers submitted from 2007 and onwards.

This returns 641 possible papers. After adding 31 additional papers through manual search and removing all duplicate results 661 papers remain. These papers are screened by their abstract and title in order to eliminate papers that are not relevant for this study. After screening only 172 papers remain and these are further filtered by their inclusion and exclusion criteria after screening the full text of the paper, leaving 117 papers (see Figure 2.1).

Independent from the systematic review, in Section 2.3 some papers are chosen to further explore techniques for the management of data in the context of MCS.

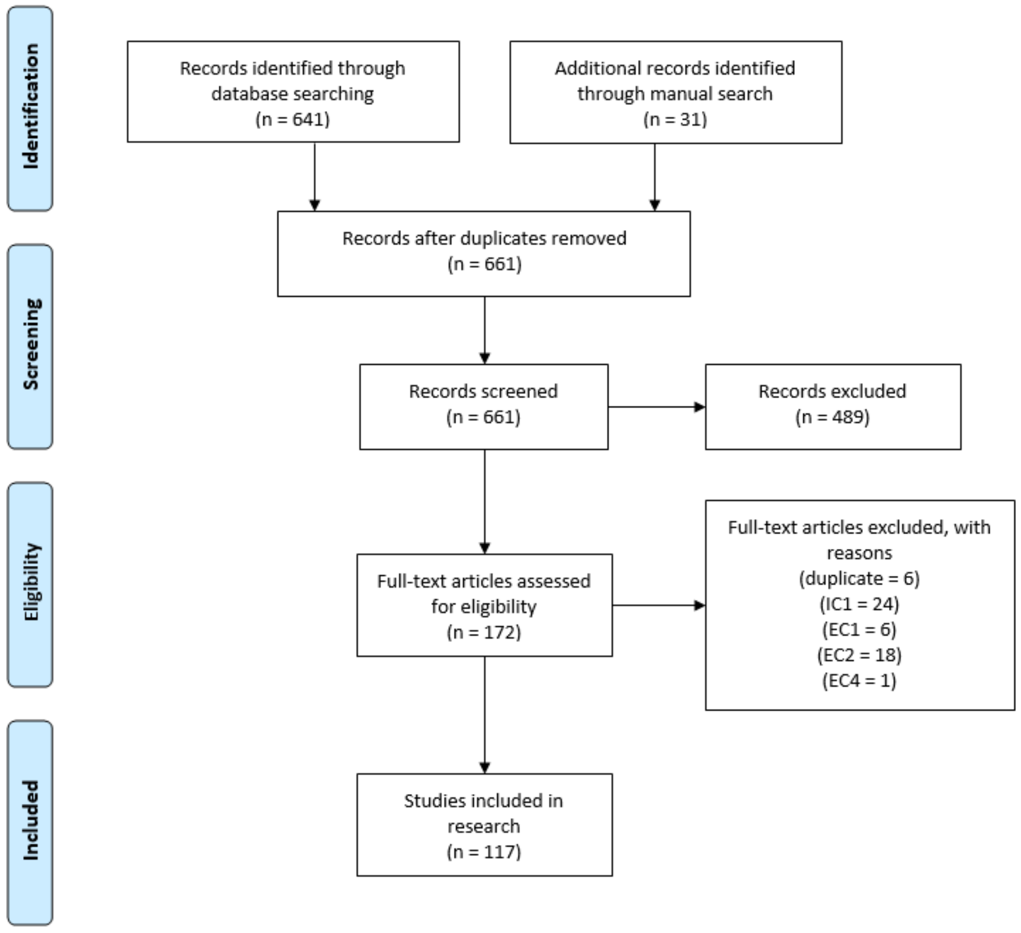

Identification
Screening
Eligibility
Included
Records identified through database searching (n = 641)
Additional records identified through manual search (n = 31)
Records after duplicates removed (n = 661)
Records screened (n = 661)
Records excluded (n = 489)
Full-text articles assessed for eligibility (n = 172)
Full-text articles excluded, with reasons (duplicate = 6) (IC1 = 24) (EC1 = 6) (EC2 = 18) (EC4 = 1)
Studies included in research (n = 117)

Figure 2.1: PRISMA [85] flow diagram of studies' screening and selection

2.2 MCS Applications - Systematic Research

This section extracts the data of the previously selected papers and compares them in regards to the following criteria:

- Goals: The goals and subgoals the paper fulfills.
- Sensor utilization: Which sensors are utilized in the paper and how/in which way/for what purpose are they used (*e.g.*, GPS used for localization or to measure the electron density in the atmosphere).
- Quality assurance: Which methods have been implemented in order to assure

the quality of information.

- Time constraint: Have there been any time constraints on the evaluation of the data in this paper (*i.e.*, were the results needed in (near) real-time?)?
- Processing device: What parts of the data processing were done on which device and why (*e.g.*, smartphone, server)?
- Evaluation method: What methods were used in order to evaluate the data? (*e.g.*, machine learning techniques)
- Reporting metric: How were the results of the paper presented (*e.g.*, accuracy)?

Please note that not necessarily all applicable papers are mentioned in the citations in order to improve the readability of the text. The full list of aspects each paper fulfills can be found in Appendix A in Table A.3 - A.11.

2.2.1 Goals

This subsection compares the different goals and subgoals of the papers. Subgoals are smaller goals the paper accomplishes (*e.g.*, *map matching* or *location matching*) and goals are used as a broader term that encompasses multiple subgoals (*e.g.*, *localization*). An arbitrary number of goals and subgoals can be achieved by each paper. For reasons of clarity and comprehensibility the papers are in this section further divided by their specific application area.

The first area, **Urban Sensing**, comprises technologies to sense and obtain data about physical areas and objects in urban spaces and how humans interact with them. This includes techniques to analyze the public infrastructure like roads [131, 49, 86] or the WiFi density of a city [37], the waiting time for specific services [148, 147, 13] and other specific applications like an online reposting system for fliers [45].

The next area, **Indoor Localization**, focuses on localization techniques for indoor environments. This is a non-trivial problem as normal localization techniques have many problems due to sensor inaccuracies within buildings leading to imprecise data. Indoor localization techniques includes localization on an indoor map [101, 146], the reconstruction of indoor maps [40, 19, 146], or other applications like generating a map of the WiFi coverage of a floor [100] and collecting fingerprints of a specific location [138].

The third area encompasses technologies regarding **Environmental Monitoring**. Environmental Monitoring is normally done by using Wireless Sensor Networks (WSN), but the installation and maintenance of these is expensive and thus MCS is often used to circumvent these costs. These applications can range from analysing nightlife behaviour of participants [108], detecting beautiful places in the city [78], and even measuring the electron count in the ionosphere [88].

The last area, **Social, Security and Healthcare (SSH)**, includes all applications regarding the physical and mental well-being of participants [95, 48], as well as applications for disaster relief [21, 123], detecting diseases [68, 31], help to observe huge crowds [132, 15, 137], letting people report events they witness [84], or determining the relationship between two people [28].

The number of papers per area can be seen in Table 2.1 and common goals of MCS-applications can be seen in Table 2.2. Furthermore in Table A.1 in Appendix A all papers can be found with their appropriate area.

	Urban	**Indoor**	**Environment**	**SSH**
Papers per area	51 (44%)	16 (14%)	19 (16%)	31 (26%)

Table 2.1: This table shows the 117 studies included in this research by their categorized area. **Urban**, **Indoor**, **Environment**, and **SSH** are used as abbreviations for the areas **Urban Sensing**, **Indoor Localization**, **Environmental Monitoring**, and **Social, Security, and Healthcare**. The number in brackets denounces the percentage of papers in that area compared to the overall number of considered papers.

Urban Sensing: The most common goal in urban sensing techniques is *localization*. *Map matching* [131, 49, 86] matches the current position to a road on existing maps. In some cases the number of possible routes can be restricted in order to have more options to achieve this goal. One specific example case can be *route matching* [148], where a list of possible routes is known. Another subgoal is simply obtaining the position of the user, *location matching* [37, 13, 147]. This knowledge can be used to *extract features at a given location* [49] or to get the *time spent at location* [147].

Another common goal is *street observation*. This includes everything relating to roads like *inferring new roads* [131], *intersection classification* [49], *detecting traffic anoma-*

	Urban	**Indoor**	**Environment**	**SSH**
Localization	49 (96%)	16 (100%)	15 (79%)	22 (71%)
Street Observation	22 (43%)	0 (0%)	1 (5%)	0 (0%)
Activity Recognition	16 (31%)	4 (25%)	1 (5%)	6 (19%)
Image Analysis	9 (18%)	3 (19%)	6 (32%)	4 (13%)
Map Generation	14 (27%)	7 (43%)	4 (21%)	1 (3%)
Sound Analysis	5 (10%)	0 (0%)	3 (16%)	4 (13%)
Air Pollution	2 (4%)	0 (0%)	7 (37%)	0 (0%)
Data Collection	2 (4%)	0 (0%)	3 (16%)	9 (29%)

Table 2.2: This table shows the goals of the papers by their categorized area. **Urban**, **Indoor**, **Environment**, and **SSH** are used as abbreviations for the areas **Urban Sensing**, **Indoor Localization**, **Environmental Monitoring**, and **Social, Security, and Healthcare**. The number in brackets denounces the percentage of papers in that area achieving this goal. The different ways *localization* was achieved can be seen in Table 2.3 below and further common goals can be seen in Table A.2 in the Appendix A.

lies [86], *determine parking spaces* [27, 133, 20], and *road surface monitoring* [6, 114].

One of the many uses of *recognition* is to reduce the amount of false data (*i.e.*, sensing data at wrong moments). Therefore *activity recognition* [13, 148] can be done, this is sometimes aided by the use of *sound recognition* [148]. *Activity recognition* is often done to help achieve other subgoals like *road surface monitoring* [6, 114] or *turn detection* [17].

	Urban	**Indoor**	**Environment**	**SSH**
Location Matching	46 (90%)	3 (19%)	14 (74%)	20 (64%)
Fingerprinting	13 (25%)	13 (81%)	1 (5%)	2 (6%)
Tracking	4 (8%)	8 (50%)	0 (0%)	0 (0%)
Event Detection	4 (8%)	1 (6%)	5 (26%)	6 (19%)

Table 2.3: This table shows in which way the papers used localization, categorized by their area. **Urban**, **Indoor**, **Environment**, and **SSH** are used as abbreviations for the areas **Urban Sensing**, **Indoor Localization**, **Environmental Monitoring**, and **Social, Security, and Healthcare**. The number in brackets denounces the percentage of papers in that area achieving *localization* in that way.

Time prediction is also often done in order to give an estimated time to enhance the user-experience. This can be done by predicting the arrival time, *arrival time prediction* [148], or the waiting time, *waiting time prediction* [147, 13], of specific services.

Many urban sensing applications aim to achieve *map generation*. These to be generated maps can range from WiFi coverage maps [37, 117, 64, 135] over cellular coverage maps [128, 38] to maps highlighting the state of the road surface [6, 114] and free parking spaces on streets [27].

Other less common subgoals of urban sensing contain *determine photo quality* [45], *photo tagging* [45], and *photo grouping* [45]. A common type of collected data is GPS traces and *splitting GPS traces* [49, 86] is sometimes done in order to analyze the data for specific information.

Indoor Localization: Again, the biggest goal in indoor localization is *localization*. As the usage of ***GPS*** in an indoor environment is very error-prone, a common way to achieve this goal is by using fingerprinting. Fingerprinting obtains the users current location by comparing current sensor readings with previously recorded sensor readings with a corresponding location. This can be done by either *WiFi-fingerprinting* [101, 100], where a list of WAPs (Wireless Access Points) and their location is stored, or *Magnetic-fingerprinting* [146], where the user needs to walk a bit in order to get the location as the magnetic fingerprints is just a 3D vector and thus needs a temporal dimension. Another way of achieving the goal is by using *tracking* [101, 100, 40, 19, 146, 90, 57, 60] using the ***accelerometer***, ***gyroscope*** and sometimes ***magnetometer*** to track the movement patterns of the user. One of these tracking-techniques is Pedestrian Dead Reckoning (PDR) [101, 100]. PDR tracks the users movement by knowing the starting location and estimating the travelled distance and direction. As PDR continuously estimates an estimation error accumulates over time and thus a combination of PDR and another indoor localization technique proves very beneficial. Simple *location matching* [19] is also done in order to detect the rough location.

Map generation is another common goal of indoor localization. *Reconstructing a floor plan* [40, 19, 146] tries to build a map of an indoor floor by using a *PDR similar approach* [146], estimating the travelled distance and direction, or by letting participants record videos or photos of the environment, which are used for *extracting information* *from picture* [40, 19] and *picture concatenation* [40], connect the adjoining wall segments of photos into continuous boundaries in order to obtain hallway connectivity, orientation and room sizes. Some MCS-applications also aim to *map WiFi coverage of an indoor floor* [100].

Other indoor localization subgoals are *navigation* [146, 90], navigating in an indoor environment, *activity recognition* [101, 100], *fingerprint collection* [138], and *QR code forgery detection* [138].

Environmental Monitoring: *Localization* is also the most common goal in environmental monitoring, but most localization-tasks in this area are pretty simple, like *location matching* [11, 26, 77, 62] is sufficient as only the location of the user is needed and *event detection* [78, 67, 140, 79], detecting the location of a physical event (*e.g.*, flowering cherry blossoms).

Detecting air pollution is one of the biggest challenges in environmental monitoring, as normal mobile phones usually do not have the necessary sensors to tackle this problem. Most applications dealing with this problem have an external mobile sensor that connects to measure the necessary data [67, 11, 62, 97, 134], but some applications try to deal with this by analysing images in order to detect pollution in the air [47, 66].

Image analysis is not only used to help with the detection of air pollution, but also for other purposes, like *analyze the brightness level of a video* [108], *analyze the loudness level of a video* [108], or simply *extracting information from picture* [78, 47, 26, 66] in order to detect a specific feature in the photos.

Other subgoals include *conduct a questionnaire* [108], *detect point of interest* [78], *expand area of interest* [78], *location matching* [78], and *measure electron count in ionosphere* [88].

Social, Security, and Healthcare (SSH): As in all other areas *localization* is the most common goal in **SSH** applications. *Location matching* [48, 28, 84, 132, 31, 50] is often required to simply get the current location of the participant. In some cases no exact location is needed, but just the knowledge if the participant is within a certain area is enough, which is called *geofencing* [15]. Also, other *event detection* [84, 68, 21, 123, 34, 56] methods like *swipe localization* [84], where multiple participants indicate a direction in which an event is taking place, are often executed.

The second most frequent goal in **SSH** applications is <u>*data collection*</u>. This goal is rather simple but can be achieved by many different methods like *conduct a questionnaire* [95, 104] or simply *data collection* [95, 9, 104]

<u>*Other*</u> subgoals of this area include *activity recognition* [48, 15], *detect nearby people* [28], *infer relationship* [28], *determine swipe direction* [84], *detect nearby Bluetooth devices* [132], and *estimate crowd density* [132].

2.2.2 Sensor Utilization

Before explaining how the sensors are used in MCS-applications the sensors are first introduced.

Sensor Introduction: Nowadays mobile phones are equipped with a multitude of sensors which can be used in order to sense all kinds of data. Most currently available mobile phones come with ***GPS***, ***gyroscope***, ***accelerometer***, ***magnetometer***, ***camera***, ***microphone***, ***fingerprint***, ***barometer***, ***ambient light***, and ***touchscreen*** [72].

The ***GPS*** [74] sensor sends signals to satellites in order to know its current distance to multiple satellites. This distance is used to calculate the exact location of the device in the form of longitude, latitude, and altitude. The ***gyroscope*** [126] measures the angular velocity around the three axis within a local coordinate system defined by the device. In a similar fashion the ***accelerometer*** [124] measures the acceleration along the three axis within a local coordinate system. The ***magnetometer*** [127] measures the strength of the magnetic field around the phone by recording the magnetic field values about the corresponding phone axis. This information can be used to obtain the phones absolute direction related to the earths geomagnetic field and thus this sensor can function as a compass. The ***camera*** can be used to take photos or videos and the ***microphone*** is necessary to record sounds. The ***fingerprint*** [72] sensor is a type of biometric recognition system and is used to identify the user. Changes in the atmospheric pressure in the surroundings are detected by the ***barometer*** [72]. The ***ambient light*** [125] sensor detects the illuminance, current ambient light level around the phone, in lux. Even the ***touchscreen*** [72] of a phone is a sensor in itself and most touchscreens distinguish between three types of input, namely tapping (clicking any location on the screen), multitouch (clicking multiple locations on the screen at once), and gesture (drawing a certain pattern on the touchscreen).

Sensor Application: The ***GPS*** sensor can be used for *map matching* [131, 49, 86] and *location matching* [147, 37, 19, 78, 48, 84, 132], but simply using ***GPS*** can have errors if the exact location is relevant and thus [147] proposed a possible solution by using the center of consecutive GPS readings. In order to *measure electron count in ionosphere* [88] ***dual-frequency GPS*** can be used. For this, ***GPS*** signals are sent on two different frequencies to the receiver and the delay between the arrival of these two signals can be used to calculate the electron count.

Another option used for *location matching* [13] is the usage of ***WiFi*** to detect WAP-locations or directly detecting a specific WAP. The ***WiFi*** sensor can also be used for *detecting WiFi density* [37], getting the currently detected WAPs, for *route matching* [148], which fingerprints cell tower IDs, and for normal *WiFi-fingerprinting* [101, 100], associating a list of WAPs with a specific location and using this for localization. ***GPS*** and ***WiFi*** can also be used together for *location matching* [28] or *geofencing* [15] to achieve even more accurate results.

The ***magnetometer*** can be used for *Magnetic-fingerprinting* [146], which works just like the WiFi equivalent with the one exception that a temporal dimension is needed, *i.e.*, the participant needs to walk the path for a little while in order to get the location. ***WiFi*** and the ***magnetometer*** can also be combined to form a combined fingerprint for *fingerprint collection* [138].

Activity recognition is most often done by using the ***accelerometer*** [148, 13, 100, 48, 15], and this can be aided by utilizing the ***microphone*** for *sound recognition* [148]. Another way to use the ***accelerometer*** is to determine the tilt angle of the phone, this information combined with the ***magnetometer*** can be used for *swipe localization* [84].

A multitude of sensors can be used for movement tracking. With just the ***gyroscope*** it can be detected if the participant makes a turn [146]. ***Accelerometer*** and ***gyroscope*** can be used together to measure distances and orientation between start and finish [40]. ***Accelerometer***, ***magnetometer*** and optionally ***gyroscope*** can all be utilized together for *PDR* [101, 100].

Other sometimes used sensors are the ***power*** sensor to detect whether a phone is charging [49], the ***camera*** can be used to make photos [45, 40] and videos [19, 108, 78], the ***microphone*** can *record ambient sound* [95], ***Bluetooth*** can *detect nearby Bluetooth devices* [132], and a combination of the ***accelerometer***, the ***magnetometer*** and the ***ambient light*** sensor can be used to *determine photo quality* [45].

2.2.3 Quality Assurance

In order to lessen the impact of outliers on the data, the accumulated data from all participants can be aggregated [19, 84], averaged [147], or clustered [131, 49]. These clusters can then be used for further computations like automatically assigning the best possible parameters [49].

This also applies to all cases of fingerprinting [101, 100, 146, 138], the more participants upload their data the better the fingerprints in the dataset get and the effect of outliers can be further reduced. This can be improved even further by choosing multiple representatives for the fingerprints by using *e.g.*, affinity propagation [146]. Localization tasks can in most cases minimize localization errors by using multiple localization techniques (*e.g.*, ***GPS*** and fingerprinting) at the same time [100, 15]. Complex localization tasks, like PDR [100], can be aided even further. Previous knowledge of the map or activity recognition can provide the opportunity to limit the error accumulation caused by noisy sensors.

In order to avoid false data, activity recognition can be done to detect if the participants are doing the to be detected activity (*e.g.*, standing in line) [148, 13]. A simple way to be able to retrace possible errors is by documenting the GPS-location error [37]. If participants are supposed to execute tasks at specific locations the current detected fingerprint can be compared to by other participants previously detected fingerprint at the same location [138] as GPS-spoofing can be easily done. If Bluetooth devices in the vicinity should be detected [132] the signal strength can also be considered in order to better be able to extrapolate the number of devices in a bigger area than the sensor can detect. To reduce the noise of sensors a median filter [48] can be applied to the sensor readings.

Manual confirmation of the participant [19, 84] can also be a good way to confirm the correctness of the data. Moreover, not all papers mentioned any steps taken in order to account for data quality [86, 45, 40, 108, 78, 95, 28].

2.2.4 Time Constraint

Most MCS-applications have no time constraints [49, 147, 45, 37, 100, 40, 19, 138] as they are used to gather non-time sensitive information, *e.g.*, to update maps or get information just for data analysis reasons.

Other MCS-applications aim to relay the gathered information, *e.g.*, estimated current waiting times, as fast as possible to the participants [148, 13, 78, 48, 15].

Many MCS-applications have a combination of real-time components and non-time sensitive components [131, 86, 101, 146]. In these cases the reason for the time constraints of the specific components is in most cases the same as mentioned above, a non-time sensitive information is required in order to further process the time sensitive information, *e.g.*, calculating the typical routing behaviour in order to detect anomalies in real-time [86]. In the case of GROPING [146], the non-time sensitive component is an update to the fingerprint map, this is at first time-sensitive as the fingerprint database needs to first be filled to execute the other components, but later on it is not important how quick this database is updated with newer fingerprints. In order to enable real-time calculation, CrowdAtlas [131] omits the clustering step in their phone application.

Table 2.4 shows the time constraints per area. It can be seen that most applications either have no time constraint at all or only some components are time relevant. Only about 20% of all the considered papers present their application as completely real-time dependant.

	Urban	**Indoor**	**Environment**	**SSH**
Real-Time	10 (20%)	1 (6%)	4 (21%)	10 (32%)
No Time Constraint	25 (49%)	10 (62%)	10 (53%)	15 (48%)
Mixed	16 (31%)	5 (31%)	5 (26%)	6 (19%)

Table 2.4: This table shows the time constraints of the applications, categorized by their area. **Urban**, **Indoor**, **Environment**, and **SSH** are used as abbreviations for the areas **Urban Sensing**, **Indoor Localization**, **Environmental Monitoring**, and **Social, Security, and Healthcare**. The number in brackets denounces the percentage of papers in that area with the corresponding time constraint.

2.2.5 Processing Device

The choice of the processing device is a really important decision and the choice made in the reviewed papers are highlighted here. The one thing all applications have in common however, is that the sensing of data is in always done by the phone or external sensors connected to the phone.

Most applications [131, 49, 86, 148, 13, 101, 19] preprocess the data locally on the phone in order to lessen the amount of data that needs to be uploaded and in order to lessen the burden on the participants phones, in cases where data transfer can be avoided at all calculations can be executed on locally if avoiding data transfer is a higher priority than avoiding computations. These preprocessing and calculations subgoals can include *route matching* [148], *splitting GPS traces* [49, 86], *sound recognition* [148], *conduct a questionnaire* [95, 108], *record ambient sound* [95], *determine swipe direction* [84], and *detect nearby Bluetooth devices* [132].

The server executed calculations are usually the more expensive calculations and can include *detecting traffic anomalies* [86], *arrival time prediction* [148], *time spent at location* [147], *waiting time prediction* [147, 13], *determine photo quality* [45], *photo tagging* [45], *photo grouping* [45], *detecting WiFi density* [37], *reconstructing a floor plan* [40, 19, 146], *picture concatenation* [40], *map WiFi coverage of an indoor floor* [100], *navigation* [146], *QR code forgery detection* [138], *analyze the brightness level of a video* [108], *analyze the loudness level of a video* [108], *detect point of interest* [78], *expand area of interest* [78], *measure electron count in ionosphere* [88], *swipe localization* [84], *detect nearby people* [28], and *estimate crowd density* [132].

Multiple subgoals are often executed on either the phone or the server. These include *map matching* [131, 49, 86], *location matching* [37, 13, 147, 19, 78, 48, 28, 84, 132], *inferring new roads* [131], *extract features at a given location* [49], *extracting information from picture* [40, 19, 78], *intersection classification* [49], *activity recognition* [13, 148, 101, 100, 48, 15], *fingerprinting* [101, 100, 146, 138], *PDR* [101, 100], and *geofencing* [15]. Especially *activity recognition* [13, 148, 101, 100, 48, 15] is most of the time executed in order to check for prerequisites for the sensing of data (*e.g.*, standing in line) and thus often done locally.

Table 2.5 shows an overview of the processing devices per area and surprisingly about 50% of the regarded MCS-applications have done no local preprocessing before uploading the data to the server. This is often the case when the main purpose of the application is *data collection* [145, 10] as that goal needs no processing, or a goal of the application is to not impede with the normal usage of the phone and thus not requiring many computational resources [100, 40, 146].

CrowdAtlas [131] uses an interesting hybrid concept where, in addition to the processing on the server after uploading the preprocessed data, all calculations can be done locally on the phone with less accuracy in order to enable real-time calculations. In addition, SmartRoad [49] tested different data aggregation schemes where they differ between calculations done locally on the phone and calculations of the aggregated data on the server. The findings were as expected, the earlier the data was aggregated (*i.e.*, uploaded to the server) the better the results.

	Urban	**Indoor**	**Environment**	**SSH**
Local Preprocess	28 (55%)	7 (44%)	8 (42%)	15 (48%)
Direct Upload	23 (45%)	9 (56%)	11 (58%)	16 (52%)

Table 2.5: This table shows the processing device of the applications, categorized by their area. **Urban**, **Indoor**, **Environment**, and **SSH** are used as abbreviations for the areas **Urban Sensing**, **Indoor Localization**, **Environmental Monitoring**, and **Social, Security, and Healthcare**. The number in brackets denounces the percentage of papers in that area with the corresponding processing device.

2.2.6 Evaluation Method

The collected data needs to be processed differently depending on the to be achieved goals and subgoals of the application. Due to the broad field of application of MCS a huge variety of techniques are utilized in MCS-applications.

One very commonly needed technique is classification. Classification techniques have multiple outcome sets and matches a new observation in the set it belongs to, *e.g.*, deciding whether something is large or small. More complex classification tasks (*e.g.*, *intersection classification* [49], *photo tagging* [45], *infer relationship* [28], ...) are often done by using existing machine learning techniques (Random Forest, AdaBoost, Support Vector Machine, ...). The less complex tasks can often be achieved without the use of machine learning and using a simple algorithm to evaluate sensor data. For *activity recognition* [148, 13, 101, 100, 48, 15] it is often enough to analyze the

accelerometer readings to discern between very few different cases, like walking vs. idle or steady motion vs. unsteady motion, and this can be easily done by simple algorithms like Signal Magnitude Area or Signal Magnitude Vector. *Sound recognition* [148] is in this regard very similar to *activity recognition* with the one exception that the per ***microphone*** recorded sound needs to be converted into frequency first, this can be done by Fast Fourier transform. If the recorded data can vary slightly each time a top-k matching algorithm, like Smith-Waterman algorithm, proves also very helpful when making classifications, *e.g.*, *route matching* [148].

Another often needed function in MCS-applications is clustering, the ability to group multiple similar objects with each other. A pretty standard use of clustering can be seen when *inferring new roads* [131]. This works by gathering multiple GPS-locations and clustering them according to Single-Linkage clustering, a hierarchical clustering approach that clusters the data points closest to each other. In order to remove duplicate images clustering can also be used for *photo grouping* [45], a widely used way to achieve this is by retrieving near duplicate images via Scale-invariant feature transform.

MCS-applications frequently ask the user to record photos or videos and therefore techniques for *image analysis* are needed. Analyzing these kinds of data often tends to be a fairly complex problem, *e.g.*, *extracting information from picture* [40, 19, 78], but thankfully there is already a huge amount of existing computer vision techniques (*e.g.*, Structure from motion, Vanishing Line Detection, Shape Matching, Wavelet Decomposition, Color Histograms, Fractal analysis, ...) that can be used to tackle these problems. There are some simpler cases like *analyze the brightness level of a video* [108], simply extract the brightness (intensity of the luminance channel) per frame and make the mean of all frames. If the distance and orientation between photos is relevant, a similar approach to *picture concatenation* [40], measuring the distance and orientation between photos and use maximum likelihood estimations to gain the relative coordinates between different photos, can be made.

Similarly *sound analysis* is also a fairly complex problem, which many MCS-applications need to tackle. In almost all cases the first step towards *sound analysis* is by converting the time domain signal to the frequency domain signal by using Fourier transform [148, 144, 50, 34]. *Speech recognition* [23] can be handled by the open source CMU Sphinx recognizer and extracting audio as features mel-frequency cepstral coefficients [23, 137] for further processing are other ways to handle sound data. When trying to *analyze* *the loudness level of a video* [108] simply extracting the audio power (calculate using the audio channel) per frame and calculate the mean of all frames is sufficient.

Most **indoor localization** techniques require the use of fingerprinting, sensing signals of the local environment and using these to fill a map of the area with the recorded signal in order to determine the current location of a user in the future. This is mostly done by utilizing ***WiFi*** for *WiFi-fingerprinting* [101, 100] or ***magnetometer*** for *Magnetic-fingerprinting* [146], these fingerprinting techniques can also be combined for an increased accuracy. A good way to use *WiFi-fingerprinting* is recording a vector of the five strongest Access Points (APs) of a location, retrieving the closest matching entry chosen by euclidean distance and returning the weighted mean of the top-k results [100]. *Magnetic-fingerprinting* works in a very similar way, the ambient magnetic field is sensed and assigned to a location on a map. However, the recorded magnetic fingerprint is just a 3D vector and while matching this vector to the map a misalignment could very easily happen due to different walking speeds. The Dynamic-Time-Warping algorithm can be used to tackle this problem, furthermore due to the nature of magnetic fingerprints the localization needs a temporal dimension [146] (*i.e.*, the users need to walk a bit). This method of localization also provides a good way to avoid and detect GPS-Spoofing, the act of altering your own GPS signal to fake your location. *QR code forgery detection* [138] uses this by comparing the currently detected fingerprints and comparing them to the centroid of previous fingerprints, should they differ too much GPS-Spoofing is suspected.

Generating maps, locating or tracking the user, and event detection is a goal of many MCS-applications. Mapping the WiFi density to a map (*detecting WiFi density* [37], *map WiFi coverage of an indoor floor* [100]) is extremely similar to the mapping process of *WiFi-fingerprinting* and is sometimes [100] even combined. In order to visualize such a coverage map the Inverse Distance Weight based spatial interpolation can be used. *Reconstructing a floor plan* [40, 19, 146] is a very complex undertaking and thus has many different approaches. A common approach is by using user recorded photos or videos and analyze these in order to gain information of the map layout [40, 19]. These information can then be used to connect the adjoining wall segments of landmarks into continuous boundaries by using combinatorical optimization and use probabilistic occupancy maps to obtain hallway connectivity, orientation and room sizes. Another approach is stitching sensed fingerprints, mapped to a segment of a map, together when parts of their segments overlaps [146]. These indoor floor plans can be used to help the user with *navigation* [146], this can be done by tracking the users current location by any means (*e.g.*, fingerprinting, PDR, ...) and using that knowledge combined with the map to execute any pathfinding algorithms, like Dijkstra's algorithm.

Other interesting, but less often used evaluation methods include the need to determine the swiping angle on the phone and methods to estimate waiting time. The swiping angle can be calculated by Equation 2.2. This calculation only works if the phone is held parallel to the ground, which can be confirmed by using the ***accelerometer*** and checking if the condition Equation 2.3 is satisfied. Waiting time can be estimated by many different approaches. One approach is to use regression to quantify the releationship between wait time and the time of day and using this to calculate a weighted euclidian distance and average the top-k closest results [13]. Another approach is recording waiting times at a specific time of day at a specific location. If there is enough data at a location simply making the average yields a solid result. But if the data is sparse at a location the knowledge from the other locations can be used and the problem can be compared to a recommender system, thus context aware collaborative filtering and factorization can be used to utilize contextual features to improve the waiting time prediction [147].

$$\Big(\text{(swiped angle on the touchscreen with respect to the phones x-axis)}- \\ \text{(angle between phones y-axis and magnetic north)}\Big) \mod 2\pi \tag{2.2}$$

$$abs(mean(acc_k)) \leq 0.05g, std(acc_k) \leq 0.1g, k \in \{x, y\} \tag{2.3}$$

2.2.7 Reporting Metric

The way the results of an MCS-application are reported is very dependent on the to be reported subgoals of the application. Subgoals that cluster, classify or simply discern between two cases (*extract features at a given location* [49], *intersection classification* [49], *route matching* [148], *activity recognition* [100, 148, 15], *sound recognition* [148], *photo grouping* [45], *photo tagging* [45], *detecting traffic anomalies* [86], *QR code forgery detection* [138], *detect point of interest* [78], *extracting* *information from picture* [78] used to classify the image, *infer relationship* [28], *estimate crowd density* [132] which estimates by classification, *geofencing* [15]) can evaluate their results by reporting their accuracy. The same principle can be applied when reporting the correctness of predicted shapes (*Reconstructing a floor plan* [40, 19]). The accuracy can be reported as Precision [49, 148, 45, 100, 40, 19], Recall [86, 45, 40, 19, 138, 78] or F-Measure [45, 40, 19]. Precision depicts the number of correct class predictions that actually belong to the correct class and Recall depicts the number of correct class predictions out of all positive outcomes, while the F-Measure provides a statistical number that regards the Precision and the Recall. The accuracy of *swipe localization* [84] was reported as reporting rate and detection rate according to Equation 2.4.

$$\begin{aligned} reportingRate &= (\#reportedEvents/\#trueEvents) \\ detectionRate &= (\#reportedTrueEvents/\#trueEvents) \end{aligned} \tag{2.4}$$

Sound recognition [148] can be further evaluated by reporting the distance to the target while reporting the accuracy and the position of the phone (*e.g.*, in hand or in bag).

All estimation techniques, like localizations (*PDR* [101, 100], *fingerprinting* [146], *location matching* [84]), extract information from video/photo (*extracting information from picture* [40], *picture concatenation* [40]), or time estimations (*arrival time prediction* [148], *waiting time prediction* [147, 13]), are best reported by the estimation error. This is most often done by simply plotting the estimation error [101, 100, 40, 146, 84] (in the respective unit, *e.g.*, meter). For more specific evaluation the mean absolute error [148, 13, 147], median absolute error [13], and the standard deviation [147] can be used. The mean absolute error represents the sum of absolute errors divided by the sample size and the median absolute error represents the median of all errors, while the standard deviation indicates how accurately the mean represents sample data.

An indirect way to evaluate a systems component is by reporting the end results with and without the usage of said component. This is often done for *activity recognition* [13]. In cases where statistical evaluation may be hard due to the lack of a ground truth, *e.g.*, generating a new map [131, 37, 100, 40, 19, 146] or service quality [146, 138, 108, 95], a manual comparison can also be made.

2.3 Advanced Techniques for Data Management

In this section different ways of managing crowdsensed data are explored. This section names some more advanced techniques that can be applicable to MCS-applications.

The network requirements for real-time data management can increase quickly and thus Sarkar *et al.* [109] propose an approach for this problem by assigning a device to a variably designed cluster. Each device sends their data to a cluster head which aggregates the data in order to reduce the transmission cost of the data to the server.

The different sensors used for mobile crowdsensing can cause a significant imbalance in the sensed data. Patel *et al.* [89] propose different solutions for this problem. The first way proposed to approach this problem is a "Data-Level Solution", balancing the data to alter the original distribution of data to achieve better classification for imbalanced datasets. The next approach is a so called "Algorithmic-Level Solution", altering the used processing algorithms to achieve better results. Finally "Cost-Sensitive and Ensemble Solutions" are explored, merges different approaches in order to achieve better accuracy and more reliable results. For each of the different solution approaches many different algorithms are presented in [89].

In crowdsensing vast amounts of data can be gathered and thus a high data dimensionality with high computing complexity can be a problem for many MCS-applications. A way to tackle this problem is by using dimension reduction techniques like Principal Component Analysis (PCA) [55]. Less dimensions in the data lead to simpler computations which makes PCA a good choice for data exploration and data preprocessing before applying more complex statistical or machine learning tools [55]. Possible advantages of PCA include:

- Obtaining better insight into the data.
- Identifying potential issues with the data such as artifacts or outliers.
- Reduce the number of predictors of a linear regression model, therefore avoiding the multicollinearity problem and reducing the risk of overfitting.
- Obtain higher accuracy and efficiency for all the methods based on computing distances between data points (e.g., K-nearest neighbors, K-means, Support Vector Machine).

- Enable the algorithm applied to the data to run faster, which provides additional time for parameters optimization or model benchmarking.
- Less storage space requirement.

The quality of data can also be a problem in MCS-applications, as the data is almost exclusively sensed in unsupervised environments. Therefore Kong *et al*. [58] provide an in-depth look at many methods to improve the quality of data. These methods include missing data reconstruction, fault data detection, data privacy preservation, multidimensional data conversion, and efficient task allocation.

Wireless Sensor networks are a vastly more explored topic than MCS and due to the similarity in terms of data collection and evaluation most techniques for data management in wireless sensor networks work for MCS. Akyildiz *et al*. [5] provide a huge in-depth analysis to wireless sensor networks and many transferable techniques including error control in transmission [5, Chapter 6], data compression before transmission and how to query sensors [5, Chapter 9], and time synchronization [5, Chapter 11] are explained in great detail.

3 Problem Approach

When considering to employ an MCS-application many options need to be regarded. This section gives an overview of the aspects to consider in the different phases an MCS-application can have. Getting the most out of your application is possible when planning the app from scratch (called **planning phase**), but this also means that the most aspects need to be considered. An already existing app can be modified to further enhance the wanted outcome after the app was deployed (called **runtime phase**). Many previously existing MCS-application have already collected vast amounts of data and not completely explored the knowledge hidden in it, thus evaluating already sensed data can also yield new insights (called **post-runtime phase**).

3.1 Planning Phase

When planning to develop an MCS-application from scratch a lot of things need to be considered, including what goals the application should fulfill, the choice of the processing device, and what methods should be taken to ensure the quality of the data and the result.

Goals: The first thing to consider is what goals and subgoals the application should fulfill and in what way these should be accomplished. Many goals need other goals or subgoals in order to be achieved, these connections are described in the following and can also be seen in Figure 3.1.

If the location of the user is needed for the application *localization* needs to be executed. This can be done in many different ways, the standard approach in big and open areas (*e.g.*, a city) is *location matching* by using ***GPS*** and in some cases ***WiFi*** or cell tower signals. If the application is supposed to work in an indoor environment

GPS is very unreliable and *fingerprinting* and *tracking* methods are preferable. In some cases the location of the user is not to be identified by coordinates but another concept, *e.g.*, is the user on a train. This can be a very application specific problem, but the most commonly used techniques to solve this problem are *activity recognition* (*e.g.*, detect the movement patterns of the specific vehicle) and *sound analysis* (*e.g.*, detect the sound of the IC card reader when boarding a bus [148]).

Street observation tries to determine different states of the road and in order to do this the current location of the participant is always relevant. Thus *localization* of some kind is always required and depending on the specific subgoals (*e.g.*, *road surface monitoring*) *activity recognition* is often required to detect the specifically observed road state.

Trying to measure *air pollution* with MCS is no easy task to fulfill by using smartphones as mobile sensors. When using only smartphone intern sensors this goal is almost always achieved by using *image analysis* to detect pollution in the air, but the standard solution is to use an external sensor that connects to the smartphone via Bluetooth. In this way even more fine-grained information regarding the air pollution, like what kind of substance is polluting the air to what degree, can be collected. This measurement is usually always coupled with the location of the sensed air pollution and thus *localization* is required.

Map generation also always needs some form of *localization* to infer the coordinates of the to be mapped objects or events. Typical application are WiFi/cellular coverage maps of cities, maps containing information generated through *street observation* or *air pollution*, or *reconstructing a floor plan*.

If the main purpose of the application is *data collection* the way to achieve this is completely dependent on the data the application wants to collect, theoretically recording all sensor readings or *conducting a questionnaire* is enough to fulfill this goal.

Activity recognition, *image analysis*, and *sound analysis* are mostly used as utility functions in order to aid another goal or subgoal of the application (see Section 2.2.6).

Processing Device: The second important consideration when designing an MCS-application is the choice of the processing device, in other words whether the application executes the calculations locally (*i.e.*, on the smartphone) or on the server. This decision can be made for the whole application as a whole, but in most cases

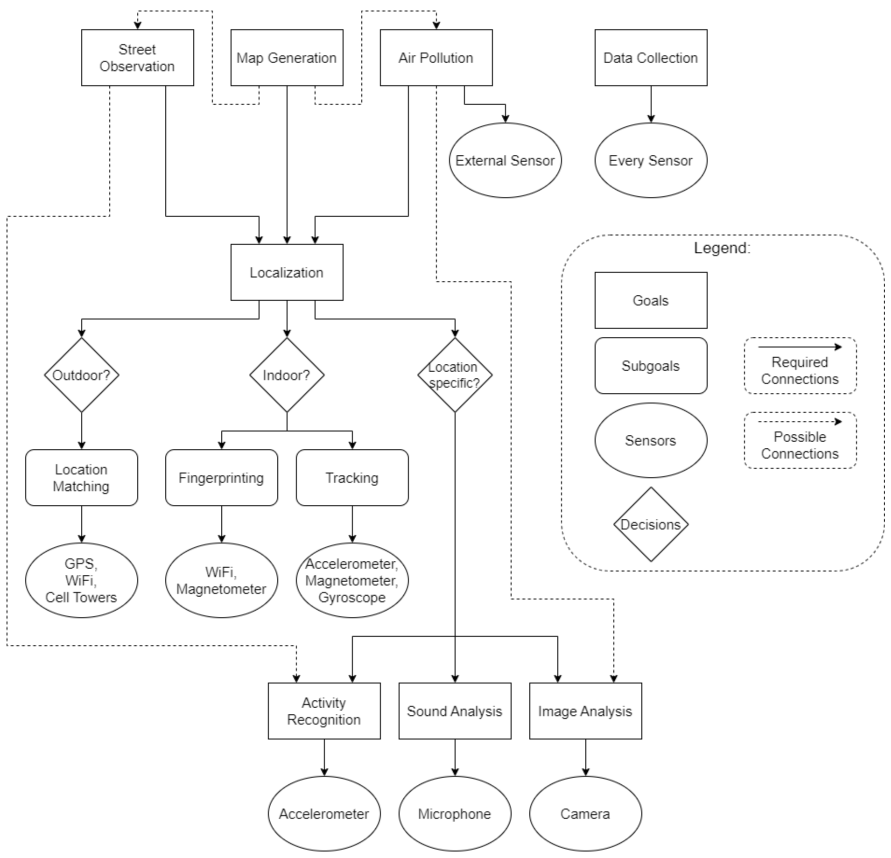


Figure 3.1: Typical goals of MCS-applications and how they commonly relate to each other. Squares represent goals, rounded squares represent subgoals, circles represent possibly needed sensors, diamonds represent decisions, and dotted lines represent common but not necessary connections. (Note that this picture only depicts common connections and not every possible connection.)

it is more sensible to make this decision for each component of the app separately as some components may have, for example, less security-relevant data or only some components are required to provide their results in real-time. In the following the most common deciding aspects for this decision are explained, which can also be seen in Figure 3.2.

The first and arguably most important aspect to consider for this decision is whether the observed component is supposed to work in (near) real-time or it has no time constraints at all. In case the results of this component are not time sensitive this aspect can be ignored. But if it tries to present the results in (near) real-time to the user the data either needs to be locally processed or a constant network connection is required (*i.e.*, no opportunistic uploading when connected to hotspots possible). Most applications are not time sensitive or only have some time sensitive parts, meaning only those parts of the application need to be especially considered when planning the processing device.

The second aspect to consider is whether the components of the application can even be feasibly executed locally. Components that only use the local data can be executed locally without many problems but components requiring data from multiple phones/users at once (*e.g.*, clustering GPS locations of multiple users) would need to download the rest of the data from the server while still uploading their own data for the other users to use. In most cases there is not much reason to do this instead of just uploading the own data to the server, letting the server make the calculations, and just downloading the calculation results from the server. If this aspect was already considered as not feasible the other decisions can be skipped for this component of the application.

Another concern is the matter of data privacy, the more data of a user is upload to the servers the more issues with data privacy can occur. Executing as many of the calculations as possible locally helps to reduce the amount of data privacy issues.

Power and resource consumption are another aspect in terms of user-friendliness. Users will not be content with the application if all the computational resources of their phones are occupied by the application and the battery life of their phones is noticeable shortened by using the application. In this aspect uploading to the server as soon as possible would be preferable, as the expensive calculations can impact the battery life noticeably [24].

Yet another aspect is the network requirements. Most users will not have an unlimited amount of mobile network data and thus considerations need to be made. If the data should possibly be uploaded anywhere and everywhere it would be preferable to execute tasks that compress the data for further calculations locally (*e.g.*, classification tasks where multiple types of data are used as input and the outcome is just a class label). Another possibility is to opportunistically upload the data when the user is connected to a WiFi hotspot, this approach circumvents the problem of data usage for the participant but does not allow for real-time results.

These aspects need to be considered for every component of the MCS-application in order to decide the processing device of the components. These decisions are everything but trivial and it boils down to a case-to-case consideration what aspects are prioritized over others.

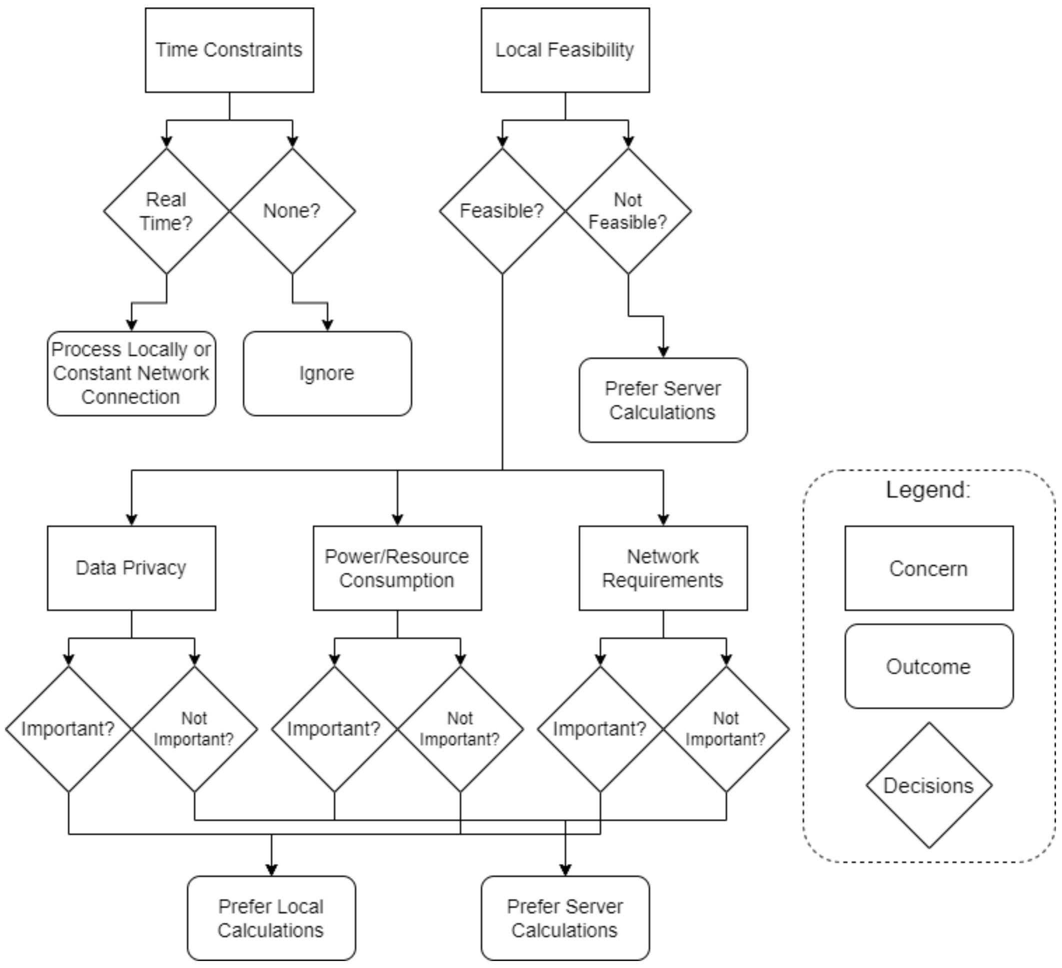


Figure 3.2: Decision diagram for deciding the processing device of a component of an MCS-application. Squares represent the considered concerns, rounded squares represent the outcome of the previous decisions, and diamonds represent decisions.

Quality Assurance: The ubiquitous deployment of mobile sensors for the use of MCS means that the data is sensed in all sorts of unsupervised situations. This provides MCS-applications with the possibility to accumulate huge amounts of data, however due to the unsupervised characteristic of the data collection process the quality of the collected data is everything but guaranteed. In order to improve or ensure the quality of the sensed data and results many precautions can be taken when developing MCS-applications. The typical pipeline for MCS-applications considers these problems by breaking them up in four major components, which can be seen in Figure 3.3. First, the data needs to be sensed in the so called *data acquisition* stage. Second, the sensed data is corrected or validated in the *data processing* stage, before missing or incorrect data is estimated in the *data imputation* stage. And then finally the gathered data from the different sensors is united and used for *data analysis*.

The first typical steps to improve the quality of the data can be applied when sensing the data in the *data acquisition* stage. Different mobile phones have different built-in sensor-hardware and therefore the sensor readings can differ from the expected values. As such, some MCS-applications provide their participants with the phones used for the sensing tasks, in order to avoid different sensor-hardware and have coherent results. This approach ensures the homogeneity of the sensed data, but reduces the potential amount of collected data greatly and increases the required budget for the project. Another possible way to improve the quality of the sensed data is by carefully selecting the participants for the data sensing tasks. By only allowing trusted participants the quality of the sensed data can be improved dramatically, but doing so greatly reduces the potential amount of collected data. In some cases the sensing of data is only desired if certain conditions are met. This condition is often the presence of the participant on a specific type of transportation vehicle, *e.g.*, train, and thus already existing *activity recognition* methods are often a good way to validate the condition.

The ubiquitousness of MCS-applications in all possible situations means that the sensed data is not always sensed correctly, under the correct conditions, or even sensed at all. For this reason the *data processing* stage focuses on validating and correcting the sensed data. For many MCS-applications the exact location of the participant is extremely important and the most commonly used technique to detect this location is *location matching* via ***GPS***. The ***GPS*** readings can potentially be quite a way off of the correct location which can cause problems for some applications. However, the severity of this problem can easily be lessened by recording the location errors as validation for the data. Besides just tracking the error many already existing methods can be applied to validate the correctness of the data such as support vector machines, artificial neural networks, Bayes classifiers, and K-nearest neighbour.

The *data imputation* stage tries to estimate missing or invalid data in order to improve the results of the application, however this approach can potentially lead to even worse results. This threat occurs most often when the amount of correct data is sparse, but due to the nature of MCS the available data should be more than enough to promise an improvement. Data imputation can be a very complex field, but luckily there already exist many techniques exactly for this problem, including Imputation Tree, K-nearest neighbour techniques and many variants (*i.e.*, K-nearest neighbour imputation, sequential K-nearest neighbour method-based imputation, and K-nearest neighbour imputation method based on Mahalanobis distance), K-means-based imputation, fuzzy C-means clustering imputation, singular value decomposition, back propagate-based neural networks, random recursive partitioning, maximum likelihood, and Bayesian estimation.

In the *data analysis* stage the sensed and corrected data is finally analyzed in the way the application wants. The here used methods are highly dependant on the application and stem most often from either machine learning, pattern recognition, or data mining. Many commonly used techniques are described in Section 2.2.6. Another often used method to improve the results of the application is by having multiple components fulfill the same goal by different subgoals, *e.g.*, in order to minimize the localization error multiple means of localization like *location matching* via ***GPS*** and *fingerprinting* can be used in conjunction. This approach leads to better, more trustworthy results, however this comes with the downside of more resource cost for the participant.

3.2 Runtime and Post-Runtime Phase

Runtime Phase: In many cases MCS-applications are already running and are supposed to be expanded. When working with an application during its runtime mostly the same aspects need to be considered as in the planning phase, but some options are lost.

The first step when trying to work with an already running MCS-application should be to explore the already collected data to get a better understanding of the already collected data.

After getting to know the dataset the next consideration should be how the application should be expanded, *i.e.*, what new goals should be achieved and can these new goals retrospectively be accomplished with the already sensed data. Figure 3.1 can be used to help with this question, as the connection between the goals and what sensors

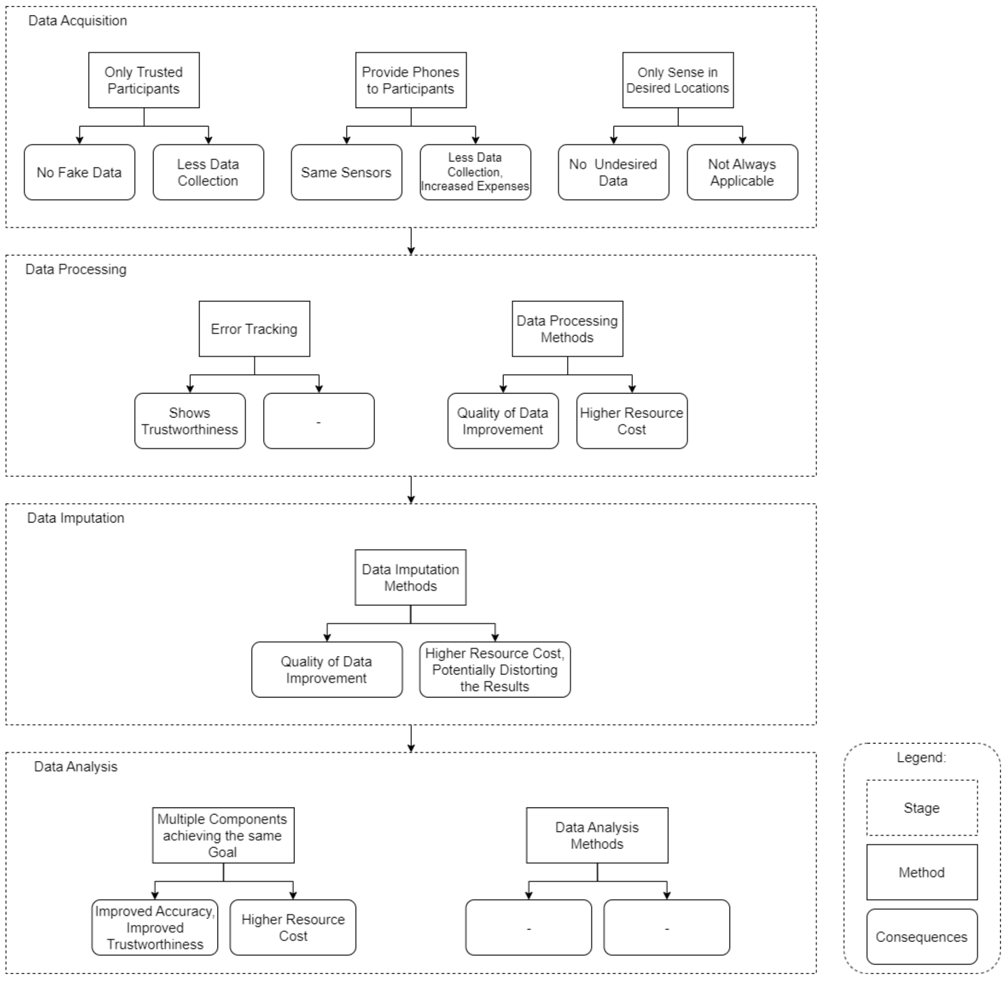

Data Acquisition
Only Trusted Participants
No Fake Data
Less Data Collection
Provide Phones to Participants
Same Sensors
Less Data Collection, Increased Expenses
Only Sense in Desired Locations
No Undesired Data
Not Always Applicable
Data Processing
Error Tracking
Shows Trustworthiness
-
Data Processing Methods
Quality of Data Improvement
Higher Resource Cost
Data Imputation
Data Imputation Methods
Quality of Data Improvement
Higher Resource Cost, Potentially Distorting the Results
Data Analysis
Multiple Components achieving the same Goal
Improved Accuracy, Improved Trustworthiness
Higher Resource Cost
Data Analysis Methods
-
-
Legend:
Stage
Method
Consequences

Figure 3.3: Typical quality assurance methods. Dotted squares represent the pipeline stages, squares represent the quality assurance methods, rounded squares represent the pros (left) and cons (right) of those methods.

they use is always the same. If the new goals cannot be achieved with the currently used sensors the application can be expanded to include more sensors in future sensing instances. When additional components are added or existing components are changed all the considerations for the processing device mentioned in Figure 3.2 need to be kept in mind.

And just like in the planning phase, quality assurance is still an important aspect. The pipeline ensuring the quality of data and the results is almost identical to the planning phase, shown in Figure 3.3. The only two differences are in the *data acquisition* stage, namely the participants can no longer be chosen and the phones can no longer be provided to ensure sensor equality, as the application is already distributed.

This shows that the steps that need to be taken to plan an MCS-application from scratch and to expand an existing one are almost identical. A step by step list of what to consider for expanding an application can be seen in Figure 3.4.

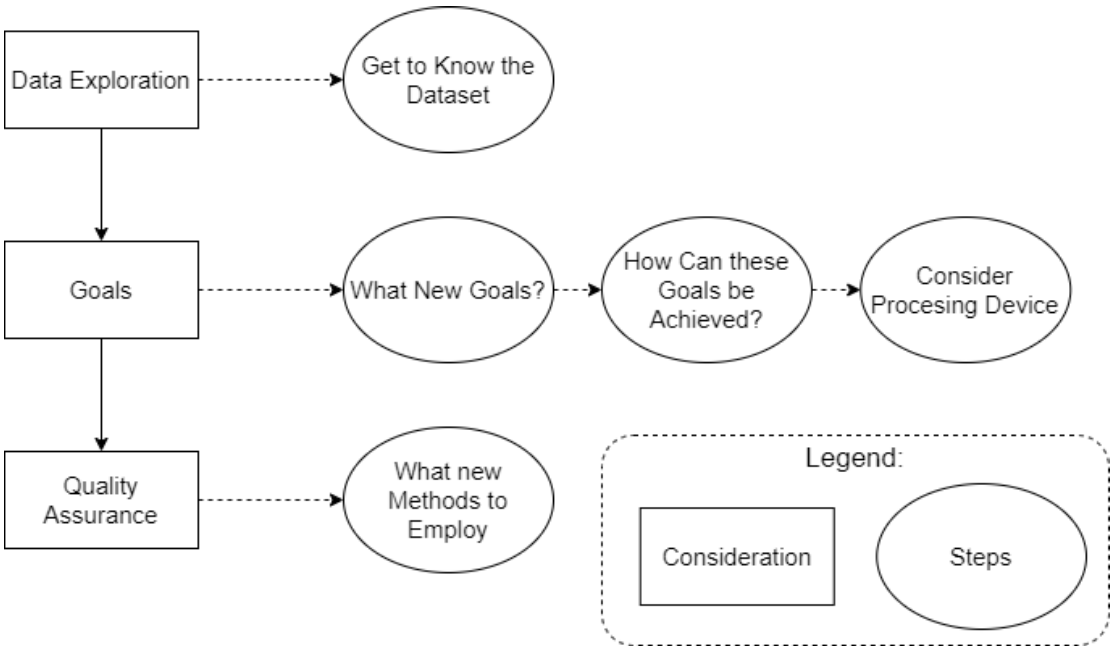


Figure 3.4: Possible considerations when expanding an MCS-application during the runtime-phase. Squares represent the considerations and circles represent the the steps taken for this consideration.

Post-Runtime Phase: During runtime MCS-applications can collect vast amounts of data from multiple sensors. These amounts of data and their connection to each other are so expansive that there is always more knowledge to be gained within the collected data. For that reason many applications do not sense at all and only use already collected data from other MCS-applications in order to analyze them even further.

When developing such a post-runtime application the considerations are almost exactly the same as for the runtime phase. The main difference is that there is no longer a *data acquisition* stage, meaning that components can no longer be added or changed to the mobile application, resulting in no new processing device considerations, the new goals need to be achieved with the already gathered data, the sensing location can no longer be restricted to improve the quality of the sensed data, and error tracking is no longer an option.

This chapter highlighted the most important aspects to consider when planning to work with MCS-applications. The most established use cases for such applications are covered and by following the mentioned steps the process of developing such an application can be somewhat streamlined. When working with an already launched or finished application it is especially important to extensively explore the already gathered data and many of the previous aspects become irrelevant.

4 Application of the Approach

In this Chapter the approach explained in Chapter 3 is demonstrated. To do so a dataset of already gathered data is used and the application of the **post-runtime phase** approach is illustrated. The used dataset is introduced and the implementation of the approach is shown.

4.1 Dataset

The used data was gathered by the TrackYourStress (TYS) application [93, 94, 52], a mHealth crowdsensing platform that is supposed to track the individual stress level of users. This is tracked by sensing the environmental sound level and the GPS position, furthermore questionnaires about the current stress level and other possibly related parameters are conducted. The progression of the TYS pipeline has two major components [93] (see Figure 4.1), the registration procedure and the continuous mobile crowdsensing procedure. The first component starts with a registration on the TYS website [52] or mobile application [94]. After that the participants have to fill in a registration questionnaire, before subscribing to the appropriate studies. The continuous mobile crowdsensing procedure starts by choosing a notification schema, which determines the frequency and types of the questionnaires. Then questionnaires popup in the given time interval and while they are filled out the application senses the environmental sound level and the GPS position. This component is looped for as long as the participants wishes to continue in the study. For the original analysis 78 participants were included and these each have conducted 4.87 questionnaires on average [93].

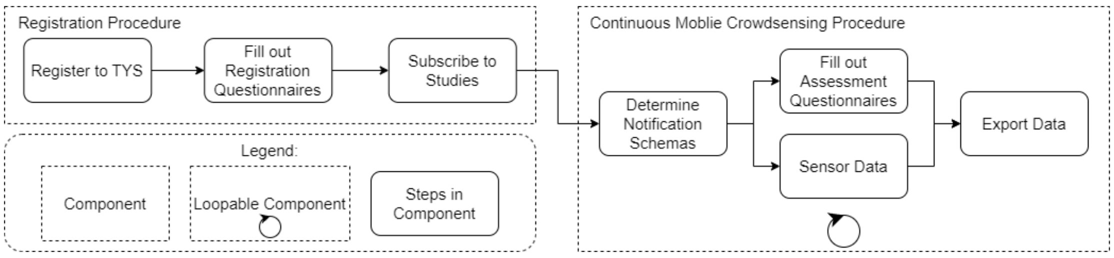

Registration Procedure
Register to TYS
Fill out Registration Questionnaires
Subscribe to Studies
Legend:
Component
Loopable Component
Steps in Component
Continuous Moblie Crowdsensing Procedure
Determine Notification Schemas
Fill out Assessment Questionnaires
Sensor Data
Export Data

Figure 4.1: TrackYourStress [93, 94, 52] progression pipeline.

4.2 Implementation

The implementation is made in the Jupyter Notebook [92] environment, a web-based integrated development environment for python $3.9.13$ [99] and offers a simple, streamlined, and document-centric experience.

The implementation follows the approach for the **post-runtime phase** as mentioned in Section 3.2. Accordingly the data is first explored, then the to be achieved goals are defined, and then the quality assurance pipeline is executed in order to achieve the results (see Figure 4.2).

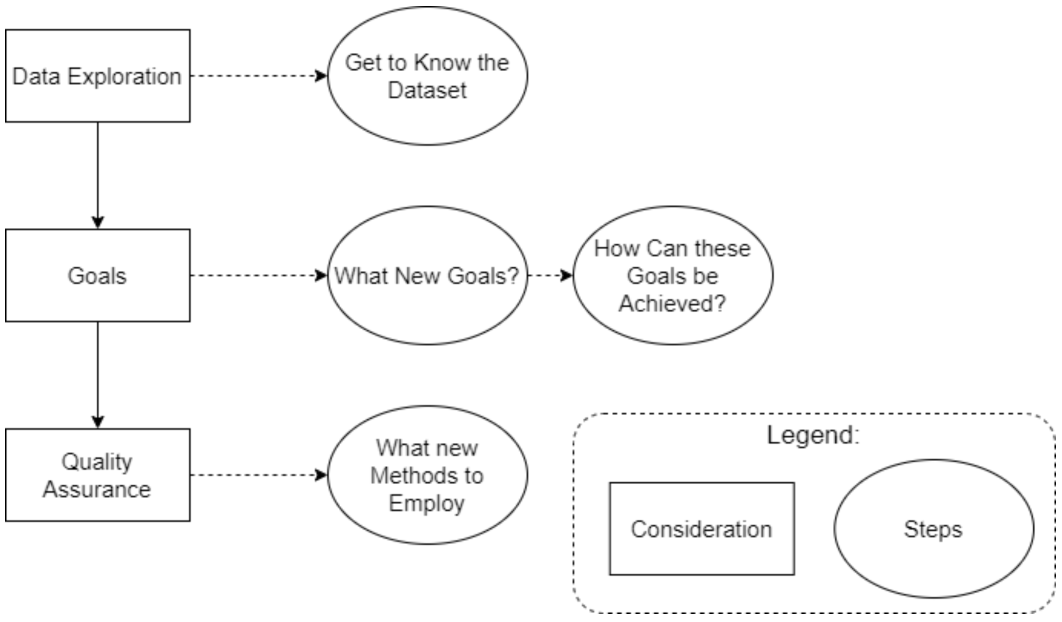
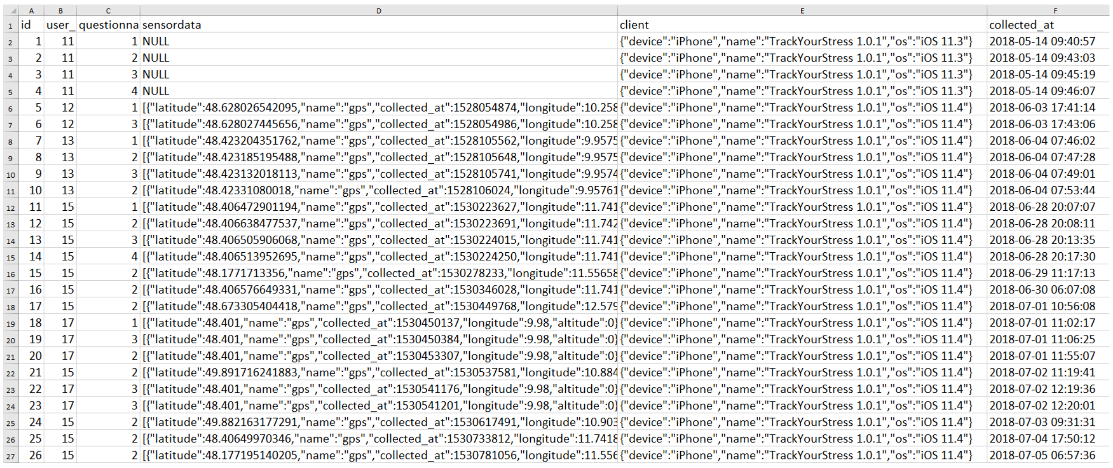


Figure 4.2: Possible considerations when expanding an MCS-application during the **post-runtime phase**. Squares represent the considerations and circles represent the the steps taken for this consideration.

Data Exploration: The data is received in the form of an Excel file, a screenshot of this file can be seen in Figure 4.3. The data contained are *id*, *user_id*, *questionnaire_id*, *sensordata*, *client*, and *collected_at*. The *id* column contains the id of the sensing run, multiple values can be sensed within a single sensing run. The *user_id* denounces the user, while the *questionnaire_id* denounces the type of questionnaire it describes. The sensed data can be found in the *sensordata* column and is in the form of an array containing multiple JSON strings, the sensed data contains ***GPS*** and ***microphone*** readings. Information about the used client can be found in a JSON string in the *client* column. And finally *collected_at* denounces the moment the questionnaire was finished.



	A	B	C	D	E	F
1	id	user_	questionna	sensordata	client	collected_at
2	1	11	1	NULL	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.3"}	2018-05-14 09:40:57
3	2	11	2	NULL	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.3"}	2018-05-14 09:43:03
4	3	11	3	NULL	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.3"}	2018-05-14 09:45:19
5	4	11	4	NULL	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.3"}	2018-05-14 09:46:07
6	5	12	1	[{"latitude":48.628026542095,"name":"gps","collected_at":1528054874,"longitude":10.258	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-03 17:41:14
7	6	12	3	[{"latitude":48.628027445656,"name":"gps","collected_at":1528054986,"longitude":10.258	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-03 17:43:06
8	7	13	1	[{"latitude":48.423204351762,"name":"gps","collected_at":1528105562,"longitude":9.9575	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-04 07:46:02
9	8	13	2	[{"latitude":48.423185195488,"name":"gps","collected_at":1528105648,"longitude":9.9575	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-04 07:47:28
10	9	13	3	[{"latitude":48.423132018113,"name":"gps","collected_at":1528105741,"longitude":9.9574	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-04 07:49:01
11	10	13	2	[{"latitude":48.42331080018,"name":"gps","collected_at":1528106024,"longitude":9.95761	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-04 07:53:44
12	11	15	1	[{"latitude":48.406472901194,"name":"gps","collected_at":1530223627,"longitude":11.741	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-28 20:07:07
13	12	15	2	[{"latitude":48.406638477537,"name":"gps","collected_at":1530223691,"longitude":11.742	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-28 20:08:11
14	13	15	3	[{"latitude":48.406505906068,"name":"gps","collected_at":1530224015,"longitude":11.741	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-28 20:13:35
15	14	15	4	[{"latitude":48.406513952695,"name":"gps","collected_at":1530224250,"longitude":11.741	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-28 20:17:30
16	15	15	2	[{"latitude":48.1771713356,"name":"gps","collected_at":1530278233,"longitude":11.55658	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-29 11:17:13
17	16	15	2	[{"latitude":48.406576649331,"name":"gps","collected_at":1530346028,"longitude":11.741	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-06-30 06:07:08
18	17	15	2	[{"latitude":48.673305404418,"name":"gps","collected_at":1530449768,"longitude":12.579	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-07-01 10:56:08
19	18	17	1	[{"latitude":48.401,"name":"gps","collected_at":1530450137,"longitude":9.98,"altitude":0}	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-07-01 11:02:17
20	19	17	3	[{"latitude":48.401,"name":"gps","collected_at":1530450384,"longitude":9.98,"altitude":0}	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-07-01 11:06:25
21	20	17	2	[{"latitude":48.401,"name":"gps","collected_at":1530453307,"longitude":9.98,"altitude":0}	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-07-01 11:55:07
22	21	15	2	[{"latitude":49.891716241883,"name":"gps","collected_at":1530537581,"longitude":10.884	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-07-02 11:19:41
23	22	17	3	[{"latitude":48.401,"name":"gps","collected_at":1530541176,"longitude":9.98,"altitude":0}	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-07-02 12:19:36
24	23	17	3	[{"latitude":48.401,"name":"gps","collected_at":1530541201,"longitude":9.98,"altitude":0}	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-07-02 12:20:01
25	24	15	2	[{"latitude":49.882163177291,"name":"gps","collected_at":1530617491,"longitude":10.903	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-07-03 09:31:31
26	25	15	2	[{"latitude":48.40649970346,"name":"gps","collected_at":1530733812,"longitude":11.7418	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-07-04 17:50:12
27	26	15	2	[{"latitude":48.177195140205,"name":"gps","collected_at":1530781056,"longitude":11.556	{"device":"iPhone","name":"TrackYourStress 1.0.1","os":"iOS 11.4"}	2018-07-05 06:57:36

Figure 4.3: Excerpt of the TYS data file.

To explore the data Python pandas version $1.5.0$ [87] is used, as it has many ready-made methods for data exploration. A quick analysis shows that there are 3070 sensing runs, 150 unique users, and 4 different values for the questionnare id. Of the sensing runs, 2933 successfully gathered data and all 3070 collected information on the client and the time of the questionnaire completion.

The contents of the client JSON string are explained in Listing 4.1 and further analysis shows that about two thirds of all the collected data was sensed by iOS devices.

Listing 4.1: Structure of the client JSON string.

```
{
 "device": "name of the device",
 "name": "version number of TYS",
 "os": "operating system on the device"
}
```

The main content modules of *sensordata* are explained in Listing 4.2. The *sensordata* column contains an array of multiple JSON strings and describes all data sensed from a single sensing run. Each JSON array contains at most one ***GPS*** reading JSON string, this reading is marked with "name" as "gps" and has a corresponding "longitude" and "latitude" value. Furthermore it also contains an "altitude" value that is sometimes also called "altidude", which is most likely just a typing mistake. Besides the ***GPS*** reading, *sensordata* also contains any number of ***microphone*** readings, which are marked with "name" as "microphone". Each of those JSON strings has a measurement of the ambient noise as either "loudness" or "amplitude". All JSON strings in *sensordata* also have an assigned "collected_at" value, marking the exact time of this sensor reading. An analysis shows that within the 3070 sensing runs only 2511 ***GPS*** readings have been recorded, but a whole of 62023 ***microphone*** readings from 2669 different sensing runs has been sensed.

Listing 4.2: Structure of the sensordata Array of JSON strings.

```
[
 {
  "name" = "gps",
  "longitude" = "longitude value",
  "latitude" = "latitude value",
  "altitude / altidude" = "altitude value",
  "collected_at" = "time stamp of this sensor reading"
 },
 {
  "name" = "microphone",
  "loudness / amplitude" = "ambient noise value",
  "collected_at" = "time stamp of this sensor reading"
 },
 ...
]
```

Goals: With the dataset explored it can now be considered what new goals should be achieved by using the existing data and how these goals should be achieved.

By following Figure 3.1 from the sensors upwards, we can see that with the ***GPS*** reading we can execute some form of *localization*, to be more specific *location matching* and with the ***microphone*** data we can apply some form of *sound analysis*. Another Possible goal that can be achieved with these inputs is *map generation*, the required *localization* is given and the data we can map is the location of the ***microphone***. Another possible goal that can be achieved would be some kind of *data analysis* that combines the location and noise level for which there are many possibilities. The form of this analysis can be anything, *e.g.*, find hotspots where the noise is notably higher than in others (which could perfectly be matched with *map generation*) or determine if there is a relation between the mean distance a person travels and the mean noise difference that person experiences. This thought process according to Figure 3.1 is visualized in Figure 4.4.

The goals chosen for this work were *map generation* in order to map the location of the microphone readings and *data analysis* to answer the question "Do people that travel a lot experience a bigger difference in the ambient noise level than people that do not?" by determining if there is a relation between the mean distance travelled and the mean noise difference per user.

Quality Assurance: The quality assurance is done according to the pipeline presented in Chapter 3, but with the according changes resulting from the **post-runtime phase**. This adjusted pipeline is shown in Figure 4.5.

Starting with the *data processing* false data is identified and corrected. For this the obvious "altidude" error is adjusted by utilizing methods from pandas [87] (see Listing 4.3). The data contained in *sensordata* is loaded into a pandas DataFrame called *sensordataValues*. All the data contained in "altidude" is copied into "altitude" and afterwards every data contained in "altidude" is deleted from the DataFrame.

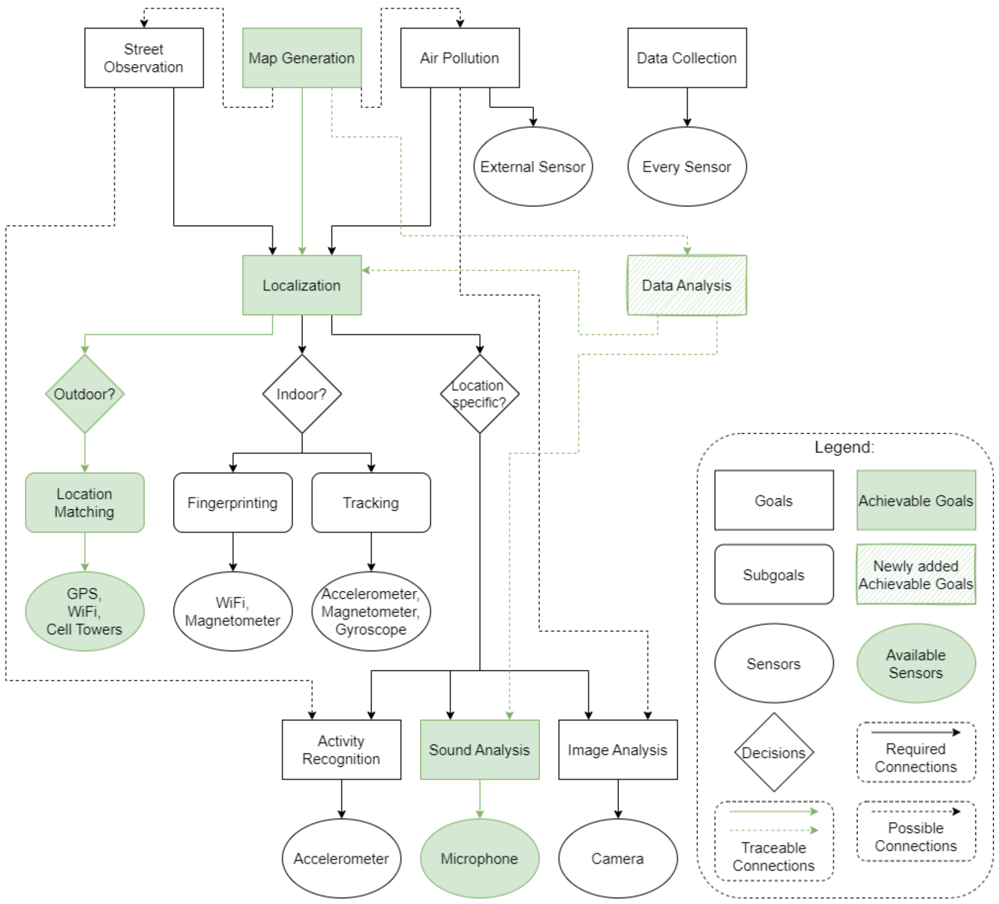


Figure 4.4: The achievable goals according to Figure 3.1. The green squares and rounded squares represent the achievable goals, the green circles show the available sensor data, and the green lines represent the traceable connections. *Data analysis* was added as this is no typical main goal, but is thinkable for this occasion.

Listing 4.3: Fixing of the "altidude" typo.

```
#Retrieve all occurrences of "altidude" and copy these
   ↪ values from "altidude" into "altitude".
sensordataValues.loc[sensordataValues["altidude"] >= 0,
   ↪ "altitude"] = sensordataValues["altidude"]

#Drop every occurrence of "altidude" from the DataFrame.
sensordataValues = sensordataValues.drop(["altidude"],
   ↪ axis = 1)
```

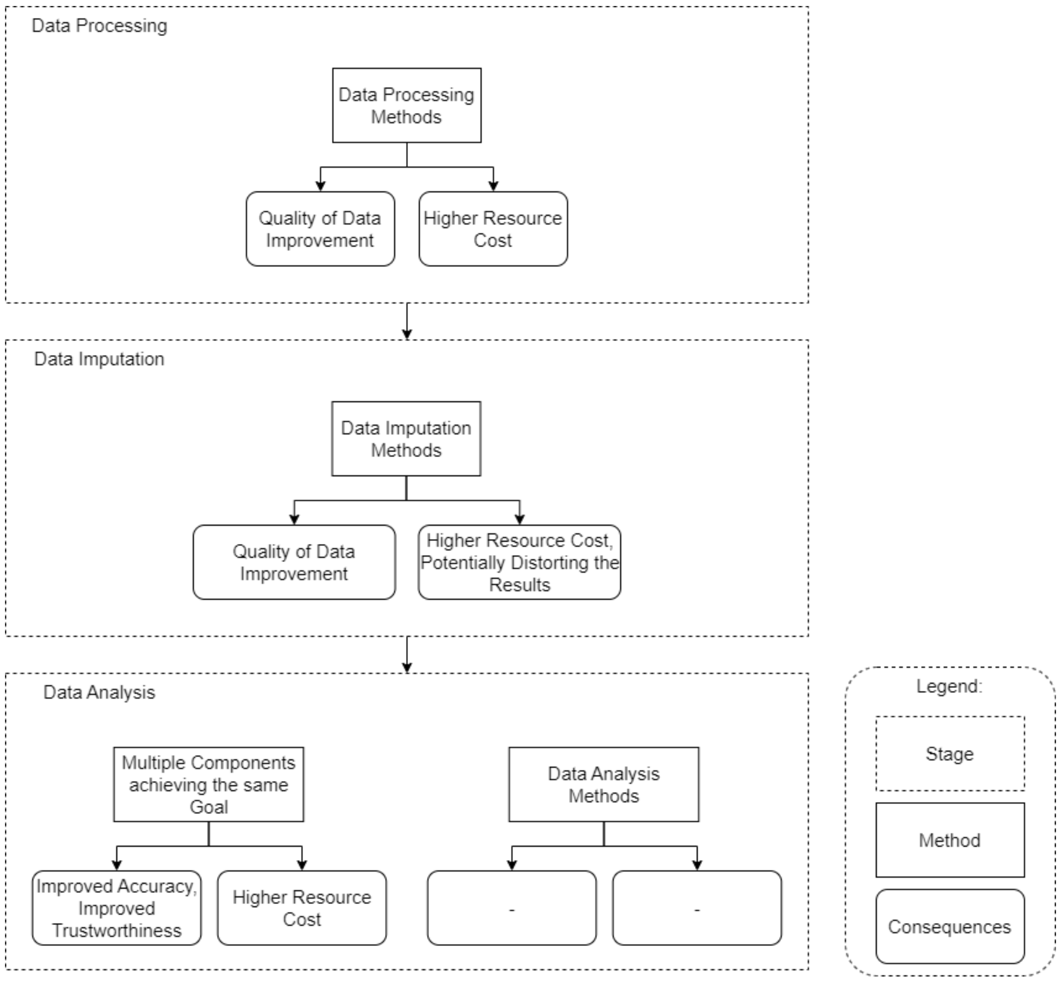

Data Processing
Data Processing Methods
Quality of Data Improvement
Higher Resource Cost
Data Imputation
Data Imputation Methods
Quality of Data Improvement
Higher Resource Cost, Potentially Distorting the Results
Data Analysis
Multiple Components achieving the same Goal
Improved Accuracy, Improved Trustworthiness
Higher Resource Cost
Data Analysis Methods
-
-
Legend:
Stage
Method
Consequences

Figure 4.5: The quality assurance pipeline according to Figure 3.3 adjusted to the **post-runtime phase**.

The next *data processing* step taken is the merging of the different ambient noise measures. The "amplitude" value denounces the recorded ambient noise as a value between 0 and 1, while "loudness" describes a value between 0 and 100. Both of these sensor readings can be interpreted as the percentage of detected volume. To merge those two, the amplitude value is multiplied by 100 and both values are copied to a new column called "noiselevel" (see Listing 4.4).

Listing 4.4: Merging "amplitude" and "loudness".

```
#A new empty list is generated.
noiseList = []

```

```
#Each sensor reading is looped and checked whether it
    ↪ contains an "amplitude" value. If such a value is
    ↪ detected it is multiplied by 100 and added to the
    ↪ list. If no such value was detected the value
    ↪ contained in "loudness" is added to the list.
for i in range (sensordataValues.shape[0]):
    if (sensordataValues["amplitude"][i] >= 0):
        noiseList.append(sensordataValues["amplitude"][i
            ↪ ] * 100)
    else:
        noiseList.append(sensordataValues["loudness"][i
            ↪ ])

#The sensordataValues DataFrame is expanded by
    ↪ "noiselevel" with the values previously contained
    ↪ in the list.
sensordataValues["noiselevel"] = noiseList
```

After that the *data processing* is done and because the amount of data available for this experiment is quite sparse for an MCS-application it is decided against any *data imputation* methods in order to avoid a distortion of the data.

At last in the *data analysis* stage the methods to analyze the data are executed. To achieve the first goal of *map generation* a list of all sensing runs containing both, a ***GPS*** and a ***microphone*** reading, is needed in order to display the locations where an ambient noise reading was made on a map. This list is then used to filter the *sensordataValues* DataFrame for these sensing runs and this is then further filtered to only contain the ***GPS*** readings in the now called *geo* DataFrame. Afterwards this DataFrame is converted to a GeoDataFrame that contains a GeometryArray created from the "longitude" and "latitude" by using geopandas version 0.11.1 [43]. This GeoDataFrame can now be plotted over a world map. The code for this process can be seen in Listing 4.5.

Listing 4.5: Generating the map with the locations of the ***microphone*** readings.

```
#Get a list with all the sensing run id's that contain
    ↪ both, GPS and microphone readings.
```

```
allSensesWithBoth = set(sensordataValues.loc[
    ↪ sensordataValues["name"] == "gps"]["id"].unique())
    ↪  & set(sensordataValues.loc[sensordataValues[
    ↪ "name"] == "microphone"]["id"].unique())
#The sensordataValued DataFrame is filtered for sensing
    ↪ runs contained in the new list.
combined = sensordataValues.loc[sensordataValues["id"].
    ↪ isin(allSensesWithBoth)]
#Further filtered to only contain GPS readings.
geo = combined.loc[combined["name"] == "gps"]
geo = geo.reset_index(drop=True)

#The DataFrame is transformed into a GeoDataFrame with
    ↪ GeometryArray created from longitude and latitude.
geoGDF = gpd.GeoDataFrame(geo, geometry=gpd.
    ↪ points_from_xy(geo.longitude, geo.latitude))

#The GeoDataFrame gets mapped over a world map
world = gpd.read_file(gpd.datasets.get_path(
    ↪ 'naturalearth_lowres'))
ax = world.plot(color='white', edgecolor='black')
geoGDF.plot(ax=ax, color='red')

plt.show()
```

The *data analysis* goal tries to find a relation between the mean distance travelled and the mean noise difference per user. To achieve this a data structure containing all the values per user is needed, therefore iterating over all users with the corresponding executed sensing tasks. The DataFrame is then filtered by the "user_id" and by iterating over all of its values a list with all the users ***GPS*** readings in the form of latitude and longitude tuples or all the users ***microphone*** readings is retrieved. By comparing all the values in this list with each other, a new list is generated containing all the distances or noise differences. This new list is then used as the value for the current user in a dictionary. The resulting dictionary is a key-value data structure containing the users as keys and their distances or noise differences as values. The process of generating this dictionary can be seen in Listing 4.6.

Listing 4.6: Generation of the distances dictionary (The noise difference dictionary is generated respectively).

```python
#Iterate over all Users with the corresponding sensor reading.
for i in userList:
    #Filter the current user and create empty lists.
    userData = location.loc[location["user_id"] == i]
    userData = userData.reset_index(drop=True)
    gpsOfUser = []
    listOfDistancesPerUser = []

    #Iterate over all readings from this user and fill the list with latitude and longitude tuples.
    for j in range (userData.shape[0]):
        gpsOfUser.append((userData.latitude[j], userData.longitude[j]))
        #print(gpsOfUser)

    #If this user has more than one reading, compare each GPS location with each other, calculate the distances of this user and add those to the dictionary. Else add an empty list in the dictionary.
    if(len(gpsOfUser) > 1):
        for j in range (len(gpsOfUser)):
            for k in range(j + 1, len(gpsOfUser)):
                listOfDistancesPerUser.append(geopy.distance.distance(gpsOfUser[j], gpsOfUser[k]).km)
        userDistanceDict[i] = listOfDistancesPerUser
    else:
        userDistanceDict[i] = []
```

After that, the mean is calculated by simply iterating over these dictionaries and calculating the mean of all the values for each specific key, this is again saved in a dictionary called *meanDistanceDict* or *meanNoiseDiffDict*. After filtering these dictionaries to only contain the keys ("user_id") occurring in both dictionaries and converting all values in a pandas Series, the data can now be analyzed. This final step before the analysis is shown in Listing 4.7.

Listing 4.7: Calculating the mean and converting to pandas Series.

```
#Iterate over the dictionary in order to create a new dictionary containing the mean of each user (only do this if the value is not an empty list).
for key, value in userNoiseDict.items():
    if(len(value) > 0):
        meanNoiseDiffDict[key] = np.mean(value)

#Filtering the dictionaries by intersecting keys.
keysInBothDicts = list(set(meanNoiseDiffDict.keys()) & set(meanDistanceDict.keys()))
meanNoiseDiffDictInstersection = {}
meanDistanceInstersection = {}
for key in keysInBothDicts:
    meanNoiseDiffDictInstersection[key] = meanNoiseDiffDict[key]
    meanDistanceInstersection[key] = meanDistanceDict[key]

#Convert to pandas Series.
noiseSeries = pd.Series(list(meanNoiseDiffDictInstersection.values()))
distanceSeries = pd.Series(list(meanDistanceInstersection.values()))
```

Finally the data is analyzed (see Listing 4.8) in the form of the covariance, by using a pandas method, and the different types of correlation, by utilizing statistical functions provided by the stats [112] module of scipy version $1.9.1$ [110]. In the end the correlation is plotted by using pyplot [98] from the matplotlib version $3.6.0$ [73].

Listing 4.8: Calculating the covariance and correlation. Plotting the correlation

```
#Calculate covariance.
meanDataFrame = pd.concat([noiseSeries, distanceSeries],
    ↪ axis = 1)
print(meanDataFrame.cov())

#Calculate the three types of correlation with P-value.
print(scipy.stats.pearsonr(noiseSeries, distanceSeries))
print(scipy.stats.spearmanr(noiseSeries, distanceSeries)
    ↪ )
print(scipy.stats.kendalltau(noiseSeries, distanceSeries
    ↪ ))

#Plot the correlation
plt.title('Correlation')
plt.scatter(noiseSeries, distanceSeries)
plt.plot(np.unique(noiseSeries), np.poly1d(np.polyfit(
    ↪ noiseSeries, distanceSeries, 1)) (np.unique(
    ↪ noiseSeries)), color='red')
plt.xlabel('mean noise diff')
plt.ylabel('mean distance')
plt.show()
```

Results: A quick analysis of the data shows that of the total 150 participants there are 112 that executed both sensor readings at one point. Furthermore there are 2247 sensing runs that contained both, one ***GPS*** and at least one ***microphone*** reading. The *map generation* result of mapping the location of these sensing runs on a world map can be seen in Figure 4.6. It shows that the absolute majority of ***microphone*** readings was done in Europe with some outliers in different parts of the world.

A total of 73 participants made at least two ***GPS*** and ***microphone*** readings while using the TYS application. This number is relevant because it is the minimum number required to calculate the mean of distances a user travelled and the mean noise differences the user experienced. By trying to find a relation between those means

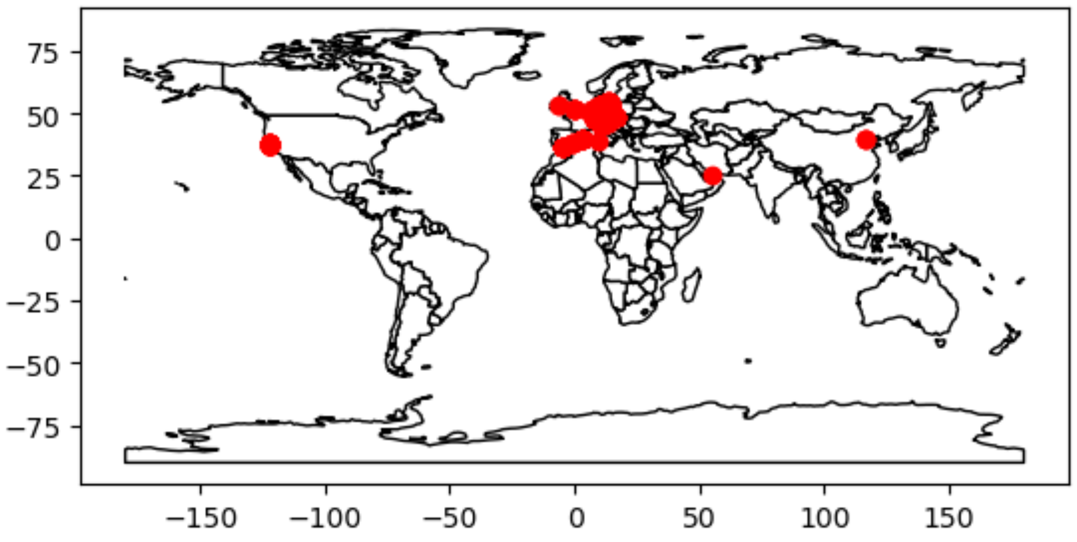


Figure 4.6: *Map generation* showing the location of all sensed ***microphone*** readings.

the *data analysis* shows a variance in the mean noise differences of 27.05% and in the mean of distances travelled of $66651.61\ km$ and a covariance between these two variables of $32.22\%km$. The positive covariance indicates that both variables move in the same direction. However, the strength of this relation cannot be inferred from this value. To get this strength, the different correlation coefficients and their P-values are calculated. All correlation coefficients are in a range between -1 and $+1$, with the sign indicating the direction and the value indicating the strength of a relation. The P-value indicates the probability of the results being a random outcome of this dataset, *i.e.*, a lower P-value denounces the trustworthiness of the results. Conventionally, for $P < 0.05$ the correlation coefficient is called statistically significant [29]. The resulting correlation is plotted in Figure 4.7. The Pearson correlation is highly influenced by outliers and thus works best for normally distributed data [81]. For this reason, the value of the Pearson correlation in this study is extremely low at 0.024 and the P-value is extremely high at 0.840. The Spearman correlation is appropriate when the variables are skewed and thus robust for outliers [81]. This results in a better correlation value of 0.209 and a P-value of 0.076. Finally, the Kendall rank correlation describes not the relation between the values of the variables, but the rank of these values [111]. This also leads to high robustness for outliers and results in a correlation value of 0.155, but with the best P-value of 0.052. All these correlation values indicate that there is a weak relation between the mean distance travelled and the mean noise difference experienced.

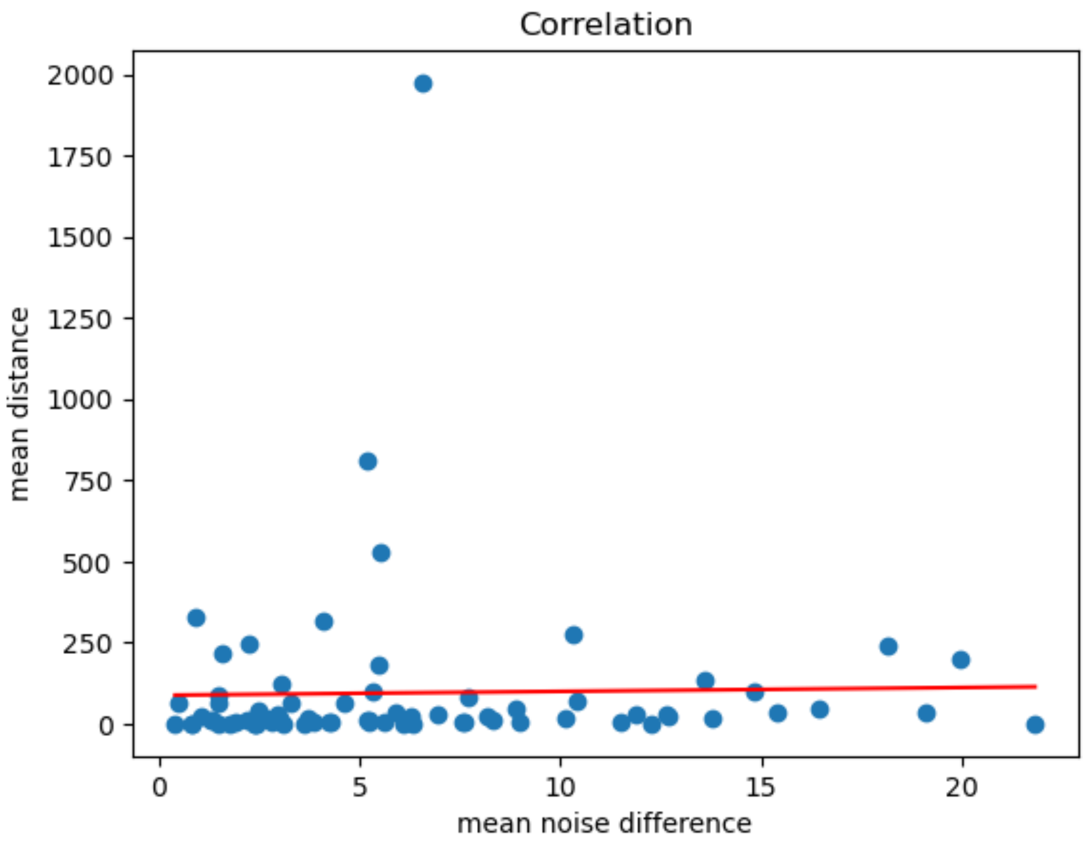


Figure 4.7: Plot showing the correlation between the mean distance travelled and the mean noise difference experienced.

This chapter visualized the application of the approach suggested in Chapter 3. The streamlined process made the strategy to develop this **post-runtime phase** MCS-application simple and helped with the execution.

5 Conclusion

MCS is an emerging topic that tries to improve the applicability and cost efficiency of conventional Wireless Sensor Networks by utilizing the sensors embedded in the mobile phones of ordinary citizens. However, most ordinary citizens will not use an application that notably drains their resources and network capacity and exposes their personal data while not gaining any benefits from it. For this reason there are many concerns when creating an MCS-application in order to minimize these problems for the users.

The first focus of this work was an exhaustive literature research on the topic of MCS-applications and compared 117 different works in terms of different key aspects. The first aspect were the goals an application fulfills. Next the sensor utilization, which sensors were utilized and in which way they were used, was examined. As the third aspect, methods ensuring the quality of information were regarded. After that, the time constraints of an application were investigated. The processing device, the device executing the calculations for the application, was regarded as the next aspect. After that, an investigation on the used evaluation methods was conducted. What reporting metrics were used in what cases was regarded as the last aspects to compare the works by. The knowledge gained through this research was used to create a streamlined approach for developing MCS-applications. This approach focuses mainly on what typical MCS goals are and how those goals are achieved, what processing device the calculations should be executed on, and how to efficiently develop the application in order to ensure the quality of the sensed data and the results. The effectiveness of this approach was then displayed by creating a new MCS-application while following the steps mentioned in it. The application was developed in python $3.9.13$ and utilizes data previously sensed by the TrackYourStress app and analyzes the connection between ***GPS*** and ***microphone*** readings.

The procedure of designing and implementing this application went smoothly and thus shows the applicability of the approach. Further research on this topic can be executed to expand this approach to other areas this work did not focus on. A few examples of these areas would be the incentive mechanism, how are ordinary citizens persuaded to participate in the application, or task allocation, how are sensing tasks efficiently allocated to the participants.

A Appendix

Area	Papers in that Area
Urban	[131, 49, 86, 148, 147, 13, 45, 37, 27, 129, 7, 30, 8, 142, 117, 23, 22, 12, 116, 36, 75, 128, 102, 38, 130, 136, 133, 139, 33, 105, 149, 6, 18, 114, 64, 76, 20, 135, 121, 25, 122, 63, 150, 32, 35, 113, 17, 120, 61, 91, 14]
Indoor	[101, 100, 40, 19, 146, 138, 90, 59, 70, 65, 103, 57, 60, 54, 119, 42]
Environment	[108, 88, 78, 67, 140, 79, 115, 144, 141, 107, 83, 47, 11, 26, 77, 62, 97, 66, 134]
SSH	[95, 48, 28, 84, 132, 15, 2, 68, 41, 82, 4, 44, 16, 145, 10, 21, 53, 137, 24, 80, 106, 9, 104, 31, 50, 143, 123, 34, 56, 3, 71]

Table A.1: This table shows the area of the application described in the papers for the systematic research. **Urban**, **Indoor**, **Environment**, and **SSH** are used as abbreviations for the areas **Urban Sensing**, **Indoor Localization**, **Environmental Monitoring**, and **Social, Security, and Healthcare**.

	Urban	**Indoor**	**Environment**	**SSH**
Navigation	4 (8%)	2 (13%)	0 (0%)	1 (3%)
Detect Nearby BT Devices	0 (0%)	0 (0%)	1 (5%)	3 (10%)
Estimate Crowd Density	2 (4%)	0 (0%)	1 (5%)	1 (3%)
Time Estimation	6 (12%)	0 (0%)	0 (0%)	0 (0%)

Table A.2: This table shows other less often occurring goals of the papers by their categorized area. **Urban**, **Indoor**, **Environment**, and **SSH** are used as abbreviations for the areas **Urban Sensing**, **Indoor Localization**, **Environmental Monitoring**, and **Social, Security, and Healthcare**. The number in brackets denounces the percentage of papers in that area achieving this goal.

	[131]	[49]	[86]	[148]	[147]	[13]	[45]	[37]	[27]	[129]	[7]	[30]	[8]	[142]
Localization	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
Street Observation	✓	✓	✓						✓	✓		✓		
Activity Recognition				✓		✓						✓		
Image Analysis							✓							
Map Generation								✓	✓					
Sound Analysis				✓										
Detect Air Pollution														
Data Collection														
Navigation														
Detect Nearby BT Devices														
Estimate Crowd Density														
Time Estimation				✓	✓	✓								
Location Matching	✓	✓	✓		✓	✓		✓	✓	✓		✓	✓	✓
Fingerprinting				✓				✓						
Tracking														
Event Detection											✓			✓
Real-Time				✓		✓			✓				✓	
No Time Constraint		✓			✓		✓	✓			✓	✓		✓
Mixed	✓		✓							✓				
Local Preprocess	✓	✓	✓	✓		✓				✓		✓	✓	
Direct Upload					✓		✓	✓	✓		✓			✓

Table A.3: This table (Table A.3 - A.11) shows the goals, localization techniques, time constraint, and processing device of each paper. (Part 1 of 9)

	[117]	[23]	[22]	[12]	[116]	[36]	[75]	[128]	[102]	[38]	[130]	[136]	[133]
Localization	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Street Observation					✓		✓		✓			✓	✓
Activity Recognition					✓	✓						✓	
Image Analysis		✓	✓						✓				
Map Generation	✓							✓		✓			
Sound Analysis		✓											
Detect Air Pollution													
Data Collection													
Navigation									✓				✓
Detect Nearby BT Devices													
Estimate Crowd Density													
Time Estimation						✓			✓				
Location Matching	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Fingerprinting	✓	✓	✓					✓			✓		
Tracking										✓			
Event Detection							✓						
Real-Time													✓
No Time Constraint		✓	✓	✓	✓			✓		✓	✓	✓	
Mixed	✓					✓	✓		✓				
Local Preprocess		✓		✓		✓			✓	✓		✓	✓
Direct Upload	✓		✓		✓		✓	✓			✓		

Table A.4: This table (Table A.3 - A.11) shows the goals, localization techniques, time constraint, and processing device of each paper. (Part 2 of 9)

	[139]	[33]	[105]	[149]	[6]	[18]	[114]	[64]	[76]	[20]	[135]	[121]	[25]	[122]
Localization	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Street Observation	✓	✓	✓	✓	✓		✓			✓				✓
Activity Recognition		✓	✓		✓		✓			✓				✓
Image Analysis	✓			✓		✓								
Map Generation					✓		✓	✓			✓			✓
Sound Analysis	✓													✓
Detect Air Pollution														
Data Collection														
Navigation		✓							✓					
Detect Nearby BT Devices														
Estimate Crowd Density												✓	✓	
Time Estimation														✓
Location Matching		✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Fingerprinting					✓			✓			✓		✓	
Tracking					✓									
Event Detection														✓
Real-Time				✓						✓		✓		
No Time Constraint	✓		✓					✓			✓		✓	✓
Mixed		✓			✓	✓	✓		✓					
Local Preprocess	✓		✓		✓	✓				✓	✓			✓
Direct Upload		✓		✓			✓	✓	✓			✓	✓	

Table A.5: This table (Table A.3 - A.11) shows the goals, localization techniques, time constraint, and processing device of each paper. (Part 3 of 9)

	[63]	[150]	[32]	[35]	[113]	[17]	[120]	[61]	[91]	[14]	[101]	[100]	[40]	[19]
Localization	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Street Observation			✓					✓		✓				
Activity Recognition	✓		✓			✓				✓	✓	✓		
Image Analysis		✓								✓			✓	✓
Map Generation	✓			✓				✓		✓		✓	✓	✓
Sound Analysis		✓												
Detect Air Pollution		✓						✓						
Data Collection									✓					
Navigation														
Detect Nearby BT Devices														
Estimate Crowd Density														
Time Estimation														
Location Matching	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓
Fingerprinting	✓			✓							✓	✓		
Tracking	✓						✓				✓	✓	✓	✓
Event Detection														
Real-Time				✓			✓							
No Time Constraint		✓	✓			✓				✓		✓	✓	✓
Mixed	✓				✓			✓	✓		✓			
Local Preprocess	✓			✓	✓		✓	✓		✓	✓			✓
Direct Upload		✓	✓			✓			✓			✓	✓	

Table A.6: This table (Table A.3 - A.11) shows the goals, localization techniques, time constraint, and processing device of each paper. (Part 4 of 9)

	[146]	[138]	[90]	[59]	[70]	[65]	[103]	[57]	[60]	[54]	[119]	[42]	[108]	[88]
Localization	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Street Observation														
Activity Recognition									✓			✓		
Image Analysis												✓	✓	
Map Generation	✓		✓	✓					✓					✓
Sound Analysis													✓	
Detect Air Pollution														
Data Collection													✓	
Navigation	✓		✓											
Detect Nearby BT Devices														
Estimate Crowd Density														
Time Estimation														
Location Matching			✓	✓										
Fingerprinting	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓		
Tracking	✓		✓					✓	✓					
Event Detection												✓		
Real-Time												✓		
No Time Constraint		✓		✓	✓		✓	✓		✓	✓		✓	✓
Mixed	✓		✓			✓			✓					
Local Preprocess			✓	✓			✓	✓	✓				✓	
Direct Upload	✓	✓			✓	✓				✓	✓	✓		✓

Table A.7: This table (Table A.3 - A.11) shows the goals, localization techniques, time constraint, and processing device of each paper. (Part 5 of 9)

	[78]	[67]	[140]	[79]	[115]	[144]	[141]	[107]	[83]	[47]	[11]	[26]	[77]	[62]
Localization	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Street Observation														
Activity Recognition														
Image Analysis	✓		✓							✓		✓		
Map Generation														✓
Sound Analysis						✓								
Detect Air Pollution		✓								✓	✓			✓
Data Collection									✓				✓	
Navigation														
Detect Nearby BT Devices								✓						
Estimate Crowd Density								✓						
Time Estimation														
Location Matching	✓	✓	✓	✓			✓	✓	✓		✓	✓	✓	✓
Fingerprinting					✓									
Tracking														
Event Detection	✓	✓	✓	✓			✓							
Real-Time	✓		✓											✓
No Time Constraint		✓		✓	✓	✓		✓	✓	✓	✓			
Mixed							✓					✓	✓	
Local Preprocess	✓		✓			✓	✓	✓						✓
Direct Upload		✓		✓	✓				✓	✓	✓	✓	✓	

Table A.8: This table (Table A.3 - A.11) shows the goals, localization techniques, time constraint, and processing device of each paper. (Part 6 of 9)

	[97]	[66]	[134]	[95]	[48]	[28]	[84]	[132]	[15]	[2]	[68]	[41]	[82]	[4]
Localization	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Street Observation	✓													
Activity Recognition	✓				✓				✓				✓	
Image Analysis		✓												✓
Map Generation	✓		✓									✓		
Sound Analysis	✓													
Detect Air Pollution	✓	✓	✓											
Data Collection				✓										
Navigation														
Detect Nearby BT Devices								✓						
Estimate Crowd Density								✓						
Time Estimation														
Location Matching	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Fingerprinting									✓					
Tracking														
Event Detection							✓				✓			
Real-Time		✓			✓				✓	✓		✓		✓
No Time Constraint				✓		✓	✓	✓			✓		✓	
Mixed	✓		✓											
Local Preprocess	✓			✓	✓	✓	✓	✓	✓	✓			✓	
Direct Upload		✓	✓								✓	✓		✓

Table A.9: This table (Table A.3 - A.11) shows the goals, localization techniques, time constraint, and processing device of each paper. (Part7 of 9)

	[44]	[16]	[145]	[10]	[21]	[53]	[137]	[24]	[80]	[106]	[9]	[104]	[31]	[50]
Localization		✓			✓				✓	✓			✓	✓
Street Observation														
Activity Recognition											✓			
Image Analysis					✓				✓					✓
Map Generation														
Sound Analysis							✓							✓
Detect Air Pollution														
Data Collection			✓	✓						✓	✓	✓		
Navigation														
Detect Nearby BT Devices	✓	✓												
Estimate Crowd Density														
Time Estimation														
Location Matching		✓			✓				✓				✓	✓
Fingerprinting										✓				
Tracking														
Event Detection					✓									
Real-Time	✓				✓									
No Time Constraint		✓	✓	✓		✓		✓				✓		
Mixed							✓		✓	✓	✓		✓	✓
Local Preprocess							✓				✓	✓		✓
Direct Upload	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	

Table A.10: This table (Table A.3 - A.11) shows the goals, localization techniques, time constraint, and processing device of each paper. (Part 8 of 9)

	[143]	[123]	[34]	[56]	[3]	[71]
Localization	✓	✓	✓	✓	✓	✓
Street Observation						
Activity Recognition		✓		✓		
Image Analysis						
Map Generation						
Sound Analysis			✓			✓
Detect Air Pollution						
Data Collection					✓	
Navigation					✓	
Detect Nearby BT Devices						
Estimate Crowd Density						
Time Estimation						
Location Matching	✓	✓		✓	✓	✓
Fingerprinting						
Tracking						
Event Detection		✓	✓	✓		
Real-Time	✓	✓	✓			
No Time Constraint				✓	✓	✓
Mixed						
Local Preprocess		✓	✓	✓		
Direct Upload	✓				✓	✓

Table A.11: This table (Table A.3 - A.11) shows the goals, localization techniques, time constraint, and processing device of each paper. (Part 9 of 9)

Bibliography

[1] *ACM Digital Library*. Accessed: 2022-09-14. URL: https://dl.acm.org/.

[2] Mahmoud A. Abdo, Ayman A. Abdel-Hamid, and Hesham A. Elzouka. "A Cloud-based Mobile Healthcare Monitoring Framework with Location Privacy Preservation". In: *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*. 2020, pp. 1–8. DOI: 10.1109/3ICT51146.2020.9311999.

[3] Dhruv Agarwal et al. "Understanding Driver-Passenger Interactions in Vehicular Crowdsensing". In: *Proc. ACM Hum.-Comput. Interact.* 5.CSCW2 (2021). DOI: 10.1145/3479869. URL: https://doi.org/10.1145/3479869.

[4] Adrian Aiordăchioae, Daniel Furtună, and Radu-Daniel Vatavu. "Aggregating Life Tags for Opportunistic Crowdsensing with Mobile and Smartglasses Users". In: *Proceedings of the 6th EAI International Conference on Smart Objects and Technologies for Social Good*. GoodTechs '20. Antwerp, Belgium: Association for Computing Machinery, 2020, 66–71. ISBN: 9781450375597. DOI: 10.1145/3411170.3411237. URL: https://doi.org/10.1145/3411170.3411237.

[5] Ian F Akyildiz and Mehmet Can Vuran. *Wireless sensor networks*. John Wiley & Sons, 2010. ISBN: 978-0-470-03601-3.

[6] Heba Aly, Anas Basalamah, and Moustafa Youssef. "Map++: A crowd-sensing system for automatic map semantics identification". In: *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 2014, pp. 546–554. DOI: 10.1109/SAHCN.2014.6990394.

[7] Theodoros Anagnostopoulos, Theodoros Xanthopoulos, and Yannis Psaromiligkos. "A Smartphone Crowdsensing System Enabling Environmental Crowdsourcing for Municipality Resource Allocation with LSTM Stochastic Prediction". In: *Sensors (Basel)* 20.14 (2020). DOI: https://doi.org/10.3390/s20143966.

[8] Luka Baljak et al. "A system for crowdsensing vibration in public transportation". In: *2019 International Conference on Artificial Intelligence: Applications and Innovations (IC-AIAI)*. 2019, pp. 30–304. DOI: `10.1109/IC-AIAI48757.2019.00012`.

[9] Erin K. Barrett et al. "Mobile Sensing: Leveraging Machine Learning for Efficient Human Behavior Modeling". In: *2020 Systems and Information Engineering Design Symposium (SIEDS)*. 2020, pp. 1–7. DOI: `10.1109/SIEDS49339.2020.9106648`.

[10] Felix Beierle et al. "Corona Health-A Study- and Sensor-Based Mobile App Platform Exploring Aspects of the COVID-19 Pandemic". In: *Int J Environ Res Public Health* 18.14 (July 2021). DOI: `https://doi.org/10.3390/ijerph18147395`.

[11] Michael Bosello, Giovanni Delnevo, and Silvia Mirri. "On Exploiting Gamification for the Crowdsensing of Air Pollution: A Case Study on a Bicycle-Based System". In: *Proceedings of the 6th EAI International Conference on Smart Objects and Technologies for Social Good*. GoodTechs '20. Antwerp, Belgium: Association for Computing Machinery, 2020, 205–210. ISBN: 9781450375597. DOI: `10.1145/3411170.3411256`. URL: `https://doi.org/10.1145/3411170.3411256`.

[12] Raj Bridgelall. "Characterizing Ride Quality With a Composite Roughness Index". In: *IEEE Transactions on Intelligent Transportation Systems* (2022), pp. 1–10. DOI: `10.1109/TITS.2021.3140177`.

[13] Muhammed Fatih Bulut, Murat Demirbas, and Hakan Ferhatosmanoglu. "LineKing: Coffee Shop Wait-Time Monitoring Using Smartphones". In: *IEEE Transactions on Mobile Computing* 14.10 (2015), pp. 2045–2058. DOI: `10.1109/TMC.2014.2384032`.

[14] Chu Cao et al. "Walkway Discovery from Large Scale Crowdsensing". In: *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. 2018, pp. 13–24. DOI: `10.1109/IPSN.2018.00009`.

[15] Giuseppe Cardone et al. "Crowdsensing in Urban Areas for City-Scale Mass Gathering Management: Geofencing and Activity Recognition". In: *IEEE Sensors Journal* 14.12 (2014), pp. 4185–4195. DOI: `10.1109/JSEN.2014.2344023`.

[16] Chieh-ming Chang, Szu-Chuang Li, and Yennun Huang. "Building Bluetooth Beacon-Based Network for Spatial-Temporal Data Collection". In: *Proceedings of the 2016 International Conference on Communication and Information Systems*. ICCIS '16. Bangkok, Thailand: Association for Computing Machinery, 2016, 91–95. ISBN: 9781450347914. DOI: 10.1145/3023924.3023935. URL: https://doi.org/10.1145/3023924.3023935.

[17] Dongyao Chen and Kang G. Shin. "TurnsMap: Enhancing Driving Safety at Intersections with Mobile Crowdsensing and Deep Learning". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3.3 (2019). DOI: 10.1145/3351236. URL: https://doi.org/10.1145/3351236.

[18] Huihui Chen, Bin Guo, and Zhiwen Yu. "Measures to Improve Outdoor Crowdsourcing Photo Collection on Smart Phones". In: *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. 2019, pp. 907–915. DOI: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00183.

[19] Si Chen et al. "Crowd Map: Accurate Reconstruction of Indoor Floor Plans from Crowdsourced Sensor-Rich Videos". In: *2015 IEEE 35th International Conference on Distributed Computing Systems*. 2015, pp. 1–10. DOI: 10.1109/ICDCS.2015.9.

[20] Jim Cherian et al. "Poster: ParkGauge: Gauging the Congestion Level of Parking Garages with Crowdsensed Parking Characteristics". In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. SenSys '15. Seoul, South Korea: Association for Computing Machinery, 2015, 395–396. ISBN: 9781450336314. DOI: 10.1145/2809695.2817881. URL: https://doi.org/10.1145/2809695.2817881.

[21] Yoonjo Choi et al. "Critical Image Identification via Incident-Type Definition Using Smartphone Data during an Emergency: A Case Study of the 2020 Heavy Rainfall Event in Korea". In: *Sensors (Basel)* 21.10 (2021). DOI: https://doi.org/10.3390/s21103562.

[22] Yohan Chon, Yunjong Kim, and Hojung Cha. "Autonomous place naming system using opportunistic crowdsensing and knowledge from crowdsourcing". In:

2013 ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN). 2013, pp. 19–30. DOI: 10.1145/2461381.2461388.

[23] Yohan Chon et al. "Automatically Characterizing Places with Opportunistic Crowdsensing Using Smartphones". In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. UbiComp '12. Pittsburgh, Pennsylvania: Association for Computing Machinery, 2012, 481–490. ISBN: 9781450312240. DOI: 10.1145/2370216.2370288. URL: https://doi.org/10.1145/2370216.2370288.

[24] Yohan Chon et al. "Crowdsensing-Based Smartphone Use Guide for Battery Life Extension". In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '16. Heidelberg, Germany: Association for Computing Machinery, 2016, 958–969. ISBN: 9781450344616. DOI: 10.1145/2971648.2971728. URL: https://doi.org/10.1145/2971648.2971728.

[25] Yohan Chon et al. "Sensing WiFi Packets in the Air: Practicality and Implications in Urban Mobility Monitoring". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '14. Seattle, Washington: Association for Computing Machinery, 2014, 189–200. ISBN: 9781450329682. DOI: 10.1145/2632048.2636066. URL: https://doi.org/10.1145/2632048.2636066.

[26] Andrea Coletta et al. "Optimal Deployment in Crowdsensing for Plant Disease Diagnosis in Developing Countries". In: *IEEE Internet of Things Journal* 9.9 (2022), pp. 6359–6373. DOI: 10.1109/JIOT.2020.3002332.

[27] Vladimir Coric and Marco Gruteser. "Crowdsensing Maps of On-street Parking Spaces". In: *2013 IEEE International Conference on Distributed Computing in Sensor Systems*. 2013, pp. 115–122. DOI: 10.1109/DCOSS.2013.15.

[28] Justin Cranshaw et al. "Bridging the Gap between Physical Location and Online Social Networks". In: *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*. UbiComp '10. Copenhagen, Denmark: Association for Computing Machinery, 2010, 119–128. ISBN: 9781605588438. DOI: 10.1145/1864349.1864380. URL: https://doi.org/10.1145/1864349.1864380.

[29] Matt David. *Fundamentals of analysis*. Accessed: 2022-10-21. URL: https://dataschool.com/fundamentals-of-analysis/.

[30] Salvatore Distefano et al. "A Stack4Things-based platform for mobile crowd-sensing services". In: *2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*. 2016, pp. 1–8. DOI: `10.1109/ITU-WT.2016.7805722`.

[31] Thierry Edoh. "Risk Prevention of Spreading Emerging Infectious Diseases Using a HybridCrowdsensing Paradigm, Optical Sensors, and Smartphone". In: *J Med Syst* 42.5 (2018), p. 91. DOI: `https://doi.org/10.1007/s10916-018-0937-2`.

[32] Amr S. El-Wakeel et al. "Towards a Practical Crowdsensing System for Road Surface Conditions Monitoring". In: *IEEE Internet of Things Journal* 5.6 (2018), pp. 4672–4685. DOI: `10.1109/JIOT.2018.2807408`.

[33] Amr S. El-Wakeel et al. "iDriveSense: Dynamic Route Planning Involving Roads Quality Information". In: *2018 IEEE Global Communications Conference (GLOBECOM)*. 2018, pp. 1–6. DOI: `10.1109/GLOCOM.2018.8648009`.

[34] Viktor Erdélyi et al. "Sonoloc: Scalable Positioning of Commodity Mobile Devices". In: *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys '18. Munich, Germany: Association for Computing Machinery, 2018, 136–149. ISBN: 9781450357203. DOI: `10.1145/3210240.3210324`. URL: `https://doi.org/10.1145/3210240.3210324`.

[35] Xiaochen Fan et al. "Towards System Implementation and Data Analysis for Crowdsensing Based Outdoor RSS Maps". In: *IEEE Access* 6 (2018), pp. 47535–47545. DOI: `10.1109/ACCESS.2018.2867578`.

[36] Karoly Farkas et al. "Crowdsending based public transport information service in smart cities". In: *IEEE Communications Magazine* 53.8 (2015), pp. 158–165. DOI: `10.1109/MCOM.2015.7180523`.

[37] Arsham Farshad, Mahesh K. Marina, and Francisco Garcia. "Urban WiFi characterization via mobile crowdsensing". In: *2014 IEEE Network Operations and Management Symposium (NOMS)*. 2014, pp. 1–9. DOI: `10.1109/NOMS.2014.6838233`.

[38] Paweł Foremski et al. "Energy-Efficient Crowdsensing of Human Mobility and Signal Levels in Cellular Networks". In: *Sensors (Basel)* 15.9 (2015), pp. 22060–22088. DOI: `https://doi.org/10.3390/s150922060`.

[39] Raghu K. Ganti, Fan Ye, and Hui Lei. "Mobile crowdsensing: current state and future challenges". In: *IEEE Communications Magazine* 49.11 (2011), pp. 32–39. DOI: 10.1109/MCOM.2011.6069707.

[40] Ruipeng Gao et al. "Jigsaw: Indoor Floor Plan Reconstruction via Mobile Crowdsensing". In: *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. MobiCom '14. Maui, Hawaii, USA: Association for Computing Machinery, 2014, 249–260. ISBN: 9781450327831. DOI: 10.1145/2639108.2639134. URL: https://doi.org/10.1145/2639108.2639134.

[41] Zhigang Gao et al. "A Student Attendance Management Method Based on Crowdsensing in Classroom Environment". In: *IEEE Access* 9 (2021), pp. 31481–31492. DOI: 10.1109/ACCESS.2021.3060256.

[42] Juan A. Álvarez García et al. "Vision and Crowdsensing Technology for an Optimal Response in Security: Project results". In: *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. 2021, pp. 82–87. DOI: 10.1109/MeditCom49071.2021.9647661.

[43] *Geopandas 0.11.0¶*. Accessed: 2022-10-20. URL: https://geopandas.org/en/stable/.

[44] Avik Ghose, Chirabrata Bhaumik, and Tapas Chakravarty. "BlueEye: A System for Proximity Detection Using Bluetooth on Mobile Phones". In: *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*. UbiComp '13 Adjunct. Zurich, Switzerland: Association for Computing Machinery, 2013, 1135–1142. ISBN: 9781450322157. DOI: 10.1145/2494091.2499771. URL: https://doi.org/10.1145/2494091.2499771.

[45] Bin Guo et al. "FlierMeet: A Mobile Crowdsensing System for Cross-Space Public Information Reposting, Tagging, and Sharing". In: *IEEE Transactions on Mobile Computing* 14.10 (2015), pp. 2020–2033. DOI: 10.1109/TMC.2014.2385097.

[46] Bin Guo et al. "Mobile Crowd Sensing and Computing: The Review of an Emerging Human-Powered Sensing Paradigm". In: *ACM Comput. Surv.* 48.1 (2015). ISSN: 0360-0300. DOI: 10.1145/2794400. URL: https://doi.org/10.1145/2794400.

[47] Pengqi Hao et al. “Fine-Grained PM2.5 Detection Method based on Crowdsensing”. In: *2020 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*. 2020, pp. 1–2. DOI: 10.1109/ICCE-Taiwan49838.2020.9258279.

[48] Yi He, Ye Li, and Shu-Di Bao. “Fall detection by built-in tri-accelerometer of smartphone”. In: *Proceedings of 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics*. 2012, pp. 184–187. DOI: 10.1109/BHI.2012.6211540.

[49] Shaohan Hu et al. “SmartRoad: Smartphone-Based Crowd Sensing for Traffic Regulator Detection and Identification”. In: *ACM Trans. Sen. Netw.* 11.4 (2015). ISSN: 1550-4859. DOI: 10.1145/2770876. URL: https://doi.org/10.1145/2770876.

[50] Xiping Hu et al. “SAfeDJ: A Crowd-Cloud Codesign Approach to Situation-Aware Music Delivery for Drivers”. In: *ACM Trans. Multimedia Comput. Commun. Appl.* 12.1s (2015). ISSN: 1551-6857. DOI: 10.1145/2808201. URL: https://doi.org/10.1145/2808201.

[51] *IEEE Xplore*. Accessed: 2022-09-14. URL: https://ieeexplore.ieee.org/Xplore/home.jsp.

[52] Tinnitus Research Initiative. *Track Your Stress*. Accessed: 2022-10-20. URL: https://www.trackyourstress.org/home.

[53] Katia Jaffrès-Runser et al. “Crowdsensing mobile content and context data: Lessons learned in the wild”. In: *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2017, pp. 311–315. DOI: 10.1109/PERCOMW.2017.7917579.

[54] Ryoma Kawajiri, Masamichi Shimosaka, and Hisashi Kashima. “Steered Crowdsensing: Incentive Design towards Quality-Oriented Place-Centric Crowdsensing”. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp ’14. Seattle, Washington: Association for Computing Machinery, 2014, 691–701. ISBN: 9781450329682. DOI: 10.1145/2632048.2636064. URL: https://doi.org/10.1145/2632048.2636064.

[55] Ferath Kherif and Adeliya Latypova. "Chapter 12 - Principal component analysis". In: *Machine Learning*. Ed. by Andrea Mechelli and Sandra Vieira. Academic Press, 2020, pp. 209–225. ISBN: 978-0-12-815739-8. DOI: `https://doi.org/10.1016/B978-0-12-815739-8.00012-2`. URL: `https://www.sciencedirect.com/science/article/pii/B9780128157398000122`.

[56] Keunseo Kim et al. "TrailSense: A Crowdsensing System for Detecting Risky Mountain Trail Segments with Walking Pattern Analysis". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1.3 (2017). DOI: `10.1145/3131893`. URL: `https://doi.org/10.1145/3131893`.

[57] Yungeun Kim, Yohan Chon, and Hojung Cha. "Mobile Crowdsensing Framework for a Large-Scale Wi-Fi Fingerprinting System". In: *IEEE Pervasive Computing* 15.3 (2016), pp. 58–67. DOI: `10.1109/MPRV.2016.50`.

[58] Linghe Kong, Bowen Wang, and Guihai Chen. *When Compressive Sensing Meets Mobile Crowdsensing*. Springer, 2019. DOI: `https://doi.org/10.1007/978-981-13-7776-1`.

[59] Johannes Kässinger et al. "GreenMap: Approximated Filtering Towards Energy-Aware Crowdsensing for Indoor Mapping". In: *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. 2018, pp. 451–459. DOI: `10.1109/MASS.2018.00069`.

[60] Nicholas D. Lane et al. "Piggyback CrowdSensing (PCS): Energy Efficient Crowdsourcing of Mobile Sensor Data by Exploiting Smartphone App Opportunities". In: *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. SenSys '13. Roma, Italy: Association for Computing Machinery, 2013. ISBN: 9781450320276. DOI: `10.1145/2517351.2517372`. URL: `https://doi.org/10.1145/2517351.2517372`.

[61] Mohamed Laraki, Giovanni De Nunzio, and Laurent Thibault. "Vehicle speed trajectory estimation using road traffic and infrastructure information". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–7. DOI: `10.1109/ITSC45102.2020.9294337`.

[62] Chiara Leonardi et al. "SecondNose: An Air Quality Mobile Crowdsensing System". In: *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*. NordiCHI '14. Helsinki, Finland: Association for Computing Machinery, 2014, 1051–1054. ISBN: 9781450325424. DOI:

10.1145/2639189.2670273. URL: https://doi.org/10.1145/2639189.2670273.

[63] Zan Li et al. "SoiCP: A Seamless Outdoor–Indoor Crowdsensing Positioning System". In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 8626–8644. DOI: 10.1109/JIOT.2019.2921561.

[64] Emanuel Lima, Ana Aguiar, and Paulo Carvalho. "Offloading Surrogates Characterization via Mobile Crowdsensing". In: *Proceedings of the First ACM Workshop on Mobile Crowdsensing Systems and Applications*. CrowdSenSys '17. Delft, Netherlands: Association for Computing Machinery, 2017, 7–12. ISBN: 9781450355551. DOI: 10.1145/3139243.3139253. URL: https://doi.org/10.1145/3139243.3139253.

[65] Chang Liu, Chun Wang, and Juan Luo. "Large-Scale Deep Learning Framework on FPGA for Fingerprint-Based Indoor Localization". In: *IEEE Access* 8 (2020), pp. 65609–65617. DOI: 10.1109/ACCESS.2020.2985162.

[66] Liang Liu et al. "Third-Eye: A Mobilephone-Enabled Crowdsensing System for Air Quality Monitoring". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2.1 (2018). DOI: 10.1145/3191752. URL: https://doi.org/10.1145/3191752.

[67] Tong Liu et al. "ALC^2 : When Active Learning Meets Compressive Crowdsensing for Urban Air Pollution Monitoring". In: *IEEE Internet of Things Journal* 6.6 (2019), pp. 9427–9438. DOI: 10.1109/JIOT.2019.2939552.

[68] Jacob Livingston and Robert Steele. "A crowdsensing algorithm for imputing Zika outbreak location data". In: *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. 2017, pp. 334–340. DOI: 10.1109/UEMCON.2017.8249065.

[69] Huadong Ma, Dong Zhao, and Peiyan Yuan. "Opportunities in mobile crowd sensing". In: *IEEE Communications Magazine* 52.8 (2014), pp. 29–35. DOI: 10.1109/MCOM.2014.6871666.

[70] Sumudu Hasala Marakkalage et al. "Identifying Indoor Points of Interest via Mobile Crowdsensing: An Experimental Study". In: *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*. 2019, pp. 1–5. DOI: 10.1109/VTS-APWCS.2019.8851651.

[71] Sumudu Hasala Marakkalage et al. "Understanding the Lifestyle of Older Population: Mobile Crowdsensing Approach". In: *IEEE Transactions on Computational Social Systems* 6.1 (2019), pp. 82–95. DOI: 10.1109/TCSS.2018.2883691.

[72] Mohammad Masoud et al. "Sensors of Smart Devices in Internet of Everything (IoE) Era: Big Opportunities and Massive Doubts". In: *Journal of Sensors* 2019 (May 2019). DOI: 10.1155/2019/6514520.

[73] *Matplotlib - Visualization with python*. Accessed: 2022-10-20. URL: https://matplotlib.org/.

[74] Jules G. McNeff. "The global positioning system". In: *IEEE Transactions on Microwave Theory and Techniques* 50.3 (2002), pp. 645–652. DOI: 10.1109/22.989949.

[75] Jernej Mihelj et al. "Crowdsourced Traffic Event Detection and Source Reputation Assessment Using Smart Contracts". In: *Sensors (Basel)* 19.15 (2019). DOI: https://doi.org/10.3390/s19153267.

[76] Silvia Mirri, Catia Prandi, and Paola Salomoni. "Personalizing Pedestrian Accessible way-finding with mPASS". In: *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. 2016, pp. 1119–1124. DOI: 10.1109/CCNC.2016.7444946.

[77] Jayantrao Mohite et al. "RuPS: Rural participatory sensing with rewarding mechanisms for crop monitoring". In: *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. 2015, pp. 378–383. DOI: 10.1109/PERCOMW.2015.7134067.

[78] Shigeya Morishita et al. "SakuraSensor: Quasi-Realtime Cherry-Lined Roads Detection through Participatory Video Sensing by Cars". In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '15. Osaka, Japan: Association for Computing Machinery, 2015, 695–705. ISBN: 9781450335744. DOI: 10.1145/2750858.2804273. URL: https://doi.org/10.1145/2750858.2804273.

[79] Davide Moroni et al. "A mobile crowdsensing app for improved maritime security and awareness". In: *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom*

Workshops). 2022, pp. 103–105. DOI: 10.1109/PerComWorkshops53856.2022.9767516.

[80] Moad Mowafi, Fahed Awad, and Fida'a Al-Quran. "Distributed Visual Crowdsensing Framework for Area Coverage in Resource Constrained Environments". In: *Sensors (Basel)* 22.15 (2022). DOI: https://doi.org/10.3390/s22155467.

[81] M. M. Mukaka. "Statistics corner: A guide to appropriate use of correlation coefficient in medical research". eng. In: *Malawi medical journal : the journal of Medical Association of Malawi* 24.3 (2012). 23638278[pmid], pp. 69–71. ISSN: 1995-7270. URL: https://pubmed.ncbi.nlm.nih.gov/23638278.

[82] Xiaoguang Niu et al. "A hierarchical-learning-based crowdedness estimation mechanism for crowdsensing buses". In: *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*. 2017, pp. 1–8. DOI: 10.1109/PCCC.2017.8280471.

[83] Klimis S. Ntalianis, Andreas Kener, and Jahna Otterbacher. "Feelings' Rating and Detection of Similar Locations, Based on Volunteered Crowdsensing and Crowdsourcing". In: *IEEE Access* 7 (2019), pp. 90215–90229. DOI: 10.1109/ACCESS.2019.2926812.

[84] Robin Wentao Ouyang et al. "If You See Something, Swipe towards It: Crowdsourced Event Localization Using Smartphones". In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '13. Zurich, Switzerland: Association for Computing Machinery, 2013, 23–32. ISBN: 9781450317702. DOI: 10.1145/2493432.2493455. URL: https://doi.org/10.1145/2493432.2493455.

[85] *PRISMA Statement*. Accessed: 2022-09-14. URL: https://prisma-statement.org/.

[86] Bei Pan et al. "Crowd Sensing of Traffic Anomalies Based on Human Mobility and Social Media". In: *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL'13. Orlando, Florida: Association for Computing Machinery, 2013, 344–353. ISBN: 9781450325219. DOI: 10.1145/2525314.2525343. URL: https://doi.org/10.1145/2525314.2525343.

[87] *Pandas*. Accessed: 2022-10-20. URL: https://pandas.pydata.org/.

[88] Victor Pankratius et al. “Mobile crowd sensing in space weather monitoring: the mahali project”. In: *IEEE Communications Magazine* 52.8 (2014), pp. 22–28. DOI: 10.1109/MCOM.2014.6871665.

[89] Harshita Patel et al. “A review on classification of imbalanced data for wireless sensor networks”. In: *International Journal of Distributed Sensor Networks* 16.4 (2020), p. 1550147720916404. DOI: 10.1177/1550147720916404. eprint: https://doi.org/10.1177/1550147720916404. URL: https://doi.org/10.1177/1550147720916404.

[90] Lambros Petrou et al. “Demonstration Abstract: Crowdsourced Indoor Localization and Navigation with Anyplace”. In: *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*. IPSN ’14. Berlin, Germany: IEEE Press, 2014, 331–332. ISBN: 9781479931460.

[91] Catia Prandi et al. “Walking with Geo-Zombie: A pervasive game to engage people in urban crowdsourcing”. In: *2016 International Conference on Computing, Networking and Communications (ICNC)*. 2016, pp. 1–5. DOI: 10.1109/ICCNC.2016.7440545.

[92] *Project jupyter*. Accessed: 2022-10-20. URL: https://jupyter.org/.

[93] Rüdiger Pryss et al. “Exploring the Time Trend of Stress Levels While Using the Crowdsensing Mobile Health Platform, TrackYourStress, and the Influence of Perceived Stress Reactivity: Ecological Momentary Assessment Pilot Study”. In: *JMIR Mhealth Uhealth* 7.10 (2019), e13978. ISSN: 2291-5222. DOI: 10.2196/13978. URL: http://mhealth.jmir.org/2019/10/e13978/.

[94] Rüdiger Pryss et al. “Machine Learning Findings on Geospatial Data of Users from the TrackYourStress mHealth Crowdsensing Platform”. In: *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. 2019, pp. 350–355. DOI: 10.1109/IRI.2019.00061.

[95] Rüdiger Pryss et al. “Mobile Crowd Sensing in Clinical and Psychological Trials – A Case Study”. In: *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*. 2015, pp. 23–24. DOI: 10.1109/CBMS.2015.26.

[96] National Center for Biotechnology Information PubMed. *PubMed*. Accessed: 2022-09-14. URL: https://pubmed.ncbi.nlm.nih.gov/.

[97] Zuber Purahoo and Sudha Cheerkoot-Jalim. “SenseAPP: An IoT-Based Mobile Crowdsensing Application for Smart Cities”. In: *2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM)*. 2020, pp. 47–52. DOI: 10.1109/ELECOM49001.2020.9297018.

[98] *Pyplot tutorial - matplotlib 3.6.0 documentation*. Accessed: 2022-10-20. URL: https://matplotlib.org/stable/tutorials/introductory/pyplot.html.

[99] Python. *Python release python 3.9.13*. Accessed: 2022-10-21. URL: https://www.python.org/downloads/release/python-3913/.

[100] Valentin Radu, Lito Kriara, and Mahesh K. Marina. “Pazl: A mobile crowdsensing based indoor WiFi monitoring system”. In: *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*. 2013, pp. 75–83. DOI: 10.1109/CNSM.2013.6727812.

[101] Anshul Rai et al. “Zee: Zero-Effort Crowdsourcing for Indoor Localization”. In: *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*. Mobicom ’12. Istanbul, Turkey: Association for Computing Machinery, 2012, 293–304. ISBN: 9781450311595. DOI: 10.1145/2348543.2348580. URL: https://doi.org/10.1145/2348543.2348580.

[102] Tirathraj Ramburn, Deevash Badoreea, and Sudha Cheerkoot-Jalim. “DriveMU: A Real-time Road-Traffic Monitoring Android Application for Mauritius”. In: *2019 Conference on Next Generation Computing Applications (NextComp)*. 2019, pp. 1–8. DOI: 10.1109/NEXTCOMP.2019.8883623.

[103] Patrice Raveneau, Stéphane D’Alu, and Hervé Rivano. “Localisation based on Wi-Fi fingerprints: A crowdsensing approach with a device-to-device aim”. In: *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2017, pp. 321–325. DOI: 10.1109/PERCOMW.2017.7917581.

[104] Vassili Rivron et al. “Refining smartphone usage analysis by combining crowdsensing and survey”. In: *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. 2015, pp. 366–371. DOI: 10.1109/PERCOMW.2015.7134065.

[105] João G. P. Rodrigues, João P. Pereira, and Ana Aguiar. "Impact of Crowdsourced Data Quality on Travel Pattern Estimation". In: *Proceedings of the First ACM Workshop on Mobile Crowdsensing Systems and Applications*. CrowdSenSys '17. Delft, Netherlands: Association for Computing Machinery, 2017, 38–43. ISBN: 9781450355551. DOI: 10.1145/3139243.3139254. URL: https://doi.org/10.1145/3139243.3139254.

[106] Robert Rusek, Joaquim Melendez Frigola, and Joan Colomer Llinas. "Influence of occupant presence patterns on energy consumption and its relation to comfort: a case study based on sensor and crowd-sensed data". In: *Energy Sustain Soc* 12.1 (2022), p. 13. DOI: https://doi.org/10.1186/s13705-022-00336-6.

[107] Darshan Santani et al. "DrinkSense: Characterizing Youth Drinking Behavior Using Smartphones". In: *IEEE Transactions on Mobile Computing* 17.10 (2018), pp. 2279–2292. DOI: 10.1109/TMC.2018.2797901.

[108] Darshan Santani et al. "The Night is Young: Urban Crowdsourcing of Nightlife Patterns". In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '16. Heidelberg, Germany: Association for Computing Machinery, 2016, 427–438. ISBN: 9781450344616. DOI: 10.1145/2971648.2971713. URL: https://doi.org/10.1145/2971648.2971713.

[109] Joy Lal Sarkar et al. "A Novel Approach for Real-Time Data Management in Wireless Sensor Networks". In: *Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics*. Ed. by Atulya Nagar, Durga Prasad Mohapatra, and Nabendu Chaki. New Delhi: Springer India, 2016, pp. 599–607. ISBN: 978-81-322-2529-4.

[110] SciPy. *SciPy*. Accessed: 2022-10-20. URL: https://scipy.org/.

[111] SciPy. *Scipy.stats.kendalltau - SciPy v1.9.1 Manual*. Accessed: 2022-10-22. URL: https://docs.scipy.org/doc/scipy-1.9.1/reference/generated/scipy.stats.kendalltau.html.

[112] SciPy. *Statistical Functions (scipy.stats) - SciPy v1.9.1 Manual*. Accessed: 2022-10-20. URL: https://docs.scipy.org/doc/scipy-1.9.1/reference/stats.html.

[113] Lu Shao et al. "Traffic condition estimation using vehicular crowdsensing data". In: *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. 2015, pp. 1–8. DOI: 10.1109/PCCC.2015.7410307.

[114] Effrosyni Sigala, Efthimios Alepis, and Constantinos Patsakis. "Measuring the Quality of Street Surfaces in Smart Cities through Smartphone Crowdsensing". In: *2020 11th International Conference on Information, Intelligence, Systems and Applications (IISA*. 2020, pp. 1–8. DOI: 10.1109/IISA50023.2020.9284384.

[115] Phillip Smith et al. "A Novel Software Defined Radio for Practical, Mobile Crowd-sourced Spectrum Sensing". In: *IEEE Transactions on Mobile Computing* (2021), pp. 1–1. DOI: 10.1109/TMC.2021.3107409.

[116] Junji Takahashi et al. "Clustering for Road Damage Locations Obtained by Smartphone Accelerometers". In: *Proceedings of the Second International Conference on IoT in Urban Space*. Urb-IoT '16. Tokyo, Japan: Association for Computing Machinery, 2016, 89–91. ISBN: 9781450342049. DOI: 10.1145/2962735.2962741. URL: https://doi.org/10.1145/2962735.2962741.

[117] Manoop Talasila, Reza Curtmola, and Cristian Borcea. "Alien vs. Mobile user game: Fast and efficient area coverage in crowdsensing". In: *6th International Conference on Mobile Computing, Applications and Services*. 2014, pp. 65–74. DOI: 10.4108/icst.mobicase.2014.257779.

[118] Gehad Mohamed Tawfik et al. "A step by step guide for conducting a systematic review and meta-analysis with simulation data". In: *Trop Med Health* 47 (2019), p. 46. DOI: https://doi.org/10.1186/s41182-019-0165-6.

[119] Xiaohua Tian et al. "Toward a Quality-Aware Online Pricing Mechanism for Crowdsensed Wireless Fingerprints". In: *IEEE Transactions on Vehicular Technology* 67.7 (2018), pp. 5953–5964. DOI: 10.1109/TVT.2018.2805383.

[120] Yao Tong et al. "Vehicle Inertial Tracking via Mobile Crowdsensing: Experience and Enhancement". In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–13. DOI: 10.1109/TIM.2022.3156993.

[121] Gianni Tumedei et al. "Promoting a Safe Return to University Campuses during the COVID-19 Pandemic: Crowdsensing Room Occupancy". In: *Proceedings of the Conference on Information Technology for Social Good*. GoodIT '21. Roma, Italy: Association for Computing Machinery, 2021, 145–150. ISBN:

9781450384780. DOI: 10.1145/3462203.3475911. URL: https://doi.org/10.1145/3462203.3475911.

[122] Rohit Verma et al. “Smart-Phone Based Spatio-Temporal Sensing for Annotated Transit Map Generation”. In: *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL ’17. Redondo Beach, CA, USA: Association for Computing Machinery, 2017. ISBN: 9781450354905. DOI: 10.1145/3139958.3140005. URL: https://doi.org/10.1145/3139958.3140005.

[123] Aku Visuri et al. “Smartphone Detection of Collapsed Buildings during Earthquakes”. In: *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*. UbiComp ’17. Maui, Hawaii: Association for Computing Machinery, 2017, 557–562. ISBN: 9781450351904. DOI: 10.1145/3123024.3124402. URL: https://doi.org/10.1145/3123024.3124402.

[124] W3C. *Accelerometer*. Accessed: 2022-10-16. 2022. URL: https://www.w3.org/TR/accelerometer/.

[125] W3C. *Ambient light sensor*. Accessed: 2022-10-21. 2022. URL: https://www.w3.org/TR/ambient-light/.

[126] W3C. *Gyroscope*. Accessed: 2022-10-16. 2021. URL: https://www.w3.org/TR/gyroscope/.

[127] W3C. *Magnetometer*. Accessed: 2022-10-21. 2021. URL: https://www.w3.org/TR/magnetometer/.

[128] Hai Wang et al. “CSMC: Cellular Signal Map Construction via Mobile Crowdsensing”. In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5.4 (2022). DOI: 10.1145/3494959. URL: https://doi.org/10.1145/3494959.

[129] Xiaojie Wang et al. “A City-Wide Real-Time Traffic Management System: Enabling Crowdsensing in Social Internet of Vehicles”. In: *IEEE Communications Magazine* 56.9 (2018), pp. 19–25. DOI: 10.1109/MCOM.2018.1701065.

[130] Xiaoyan Wang et al. “Green Spectrum Sharing Framework in B5G Era by Exploiting Crowdsensing”. In: *IEEE Transactions on Green Communications and Networking* (2022), pp. 1–1. DOI: 10.1109/TGCN.2022.3186282.

[131] Yin Wang et al. "CrowdAtlas: Self-Updating Maps for Cloud and Personal Use". In: *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*. MobiSys '13. Taipei, Taiwan: Association for Computing Machinery, 2013, 27–40. ISBN: 9781450316729. DOI: 10.1145/2462456.2464441. URL: https://doi.org/10.1145/2462456.2464441.

[132] Jens Weppner and Paul Lukowicz. "Bluetooth based collaborative crowd density estimation with mobile phones". In: *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 2013, pp. 193–200. DOI: 10.1109/PerCom.2013.6526732.

[133] Di Wu et al. "Human as a Service: Towards Resilient Parking Search System With Sensorless Sensing". In: *IEEE Transactions on Intelligent Transportation Systems* 23.8 (2022), pp. 13863–13877. DOI: 10.1109/TITS.2021.3133713.

[134] Di Wu et al. "When Sharing Economy Meets IoT: Towards Fine-Grained Urban Air Quality Monitoring through Mobile Crowdsensing on Bike-Share System". In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4.2 (2020). DOI: 10.1145/3397328. URL: https://doi.org/10.1145/3397328.

[135] Xuangou Wu et al. "Privacy preserving RSS map generation for a crowdsensing network". In: *IEEE Wireless Communications* 22.4 (2015), pp. 42–48. DOI: 10.1109/MWC.2015.7224726.

[136] Xuan Xiao et al. "How Many Bumps in Your City? Personalized Bump Seeker With Mobile Crowdsensing". In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–12. DOI: 10.1109/TIM.2021.3135549.

[137] Chenren Xu et al. "Crowdsensing the speaker count in the wild: implications and applications". In: *IEEE Communications Magazine* 52.10 (2014), pp. 92–99. DOI: 10.1109/MCOM.2014.6917408.

[138] Qiang Xu and Rong Zheng. "MobiBee: A Mobile Treasure Hunt Game for Location-Dependent Fingerprint Collection". In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. UbiComp '16. Heidelberg, Germany: Association for Computing Machinery, 2016, 1472–1477. ISBN: 9781450344623. DOI: 10.1145/2968219.2968590. URL: https://doi.org/10.1145/2968219.2968590.

[139] Kuldeep Yadav et al. "Human sensors: Case-study of open-ended community sensing in developing regions". In: *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. 2013, pp. 389–392. DOI: 10.1109/PerComW.2013.6529523.

[140] Chaoqun Yang et al. "A Crowdsensing-Based Cyber-Physical System for Drone Surveillance Using Random Finite Set Theory". In: *ACM Trans. Cyber-Phys. Syst.* 3.4 (2019). ISSN: 2378-962X. DOI: 10.1145/3342049. URL: https://doi.org/10.1145/3342049.

[141] Guang Yang et al. "CEDAR: A Cost-Effective Crowdsensing System for Detecting and Localizing Drones". In: *IEEE Transactions on Mobile Computing* 19.9 (2020), pp. 2028–2043. DOI: 10.1109/TMC.2019.2921962.

[142] Takuro Yonezawa et al. "Accelerating Urban Science by Crowdsensing with Civil Officers". In: *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*. UbiComp '18. Singapore, Singapore: Association for Computing Machinery, 2018, 303–306. ISBN: 9781450359665. DOI: 10.1145/3267305.3267641. URL: https://doi.org/10.1145/3267305.3267641.

[143] Ana María Zambrano V et al. "SIGPRO: A Real-Time Progressive Notification System Using MQTT Bridges and Topic Hierarchy for Rapid Location of Missing Persons". In: *IEEE Access* 8 (2020), pp. 149190–149198. DOI: 10.1109/ACCESS.2020.3015183.

[144] W. Zamora et al. "Accurate Ambient Noise Assessment Using Smartphones". In: *Sensors (Basel)* 17.4 (2017).

[145] Mattia Zeni, Enrico Bignotti, and Fausto Giunchiglia. "Combining Crowdsourcing and Crowdsensing to Infer the Spatial Context". In: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2018, pp. 28–33. DOI: 10.1109/PERCOMW.2018.8480312.

[146] Chi Zhang et al. "GROPING: Geomagnetism and cROwdsensing Powered Indoor NaviGation". In: *IEEE Transactions on Mobile Computing* 14.2 (2015), pp. 387–400. DOI: 10.1109/TMC.2014.2319824.

[147] Fuzheng Zhang et al. "Sensing the Pulse of Urban Refueling Behavior". In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '13. Zurich, Switzerland: Association for Computing Machinery, 2013, 13–22. ISBN: 9781450317702. DOI: `10.1145/2493432.2493448`. URL: `https://doi.org/10.1145/2493432.2493448`.

[148] Pengfei Zhou, Yuanqing Zheng, and Mo Li. "How Long to Wait? Predicting Bus Arrival Time With Mobile Phone Based Participatory Sensing". In: *IEEE Transactions on Mobile Computing* 13.6 (2014), pp. 1228–1241. DOI: `10.1109/TMC.2013.136`.

[149] Hanwei Zhu and Sid Chi-Kin Chau. "Integrating IoT-Sensing and Crowdsensing for Privacy-Preserving Parking Monitoring". In: *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. BuildSys '21. Coimbra, Portugal: Association for Computing Machinery, 2021, 226–227. ISBN: 9781450391146. DOI: `10.1145/3486611.3492229`. URL: `https://doi.org/10.1145/3486611.3492229`.

[150] Qiuxi Zhu et al. "Spatiotemporal Scheduling for Crowd Augmented Urban Sensing". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 2018, pp. 1997–2005. DOI: `10.1109/INFOCOM.2018.8485869`.