

Cooperative Localisation on Android Devices by Utilising only Environmental Sound

by

Jacob Kamminga

University of Twente

Faculty of Electrical Engineering, Mathematics and Computer Science

Pervasive Systems Group

Author
Faculty of Electrical Engineering, Mathematics and Computer Science

Certified by
Paul Havinga
Full Professor
Thesis Supervisor

**Cooperative Localisation on Android Devices by Utilising
only Environmental Sound**

by
Jacob Kamminga

Submitted to the Faculty of Electrical Engineering, Mathematics and
Computer Science
on 11-09-2015, in partial fulfilment of the
requirements for the degree of
Master of Science

Abstract

This thesis focusses on research that gears towards Cooperative Localisation utilising only ambient sound. Sound signals can be used for Time Difference Of Arrival (TDOA) based Cooperative Localisation on mobile devices. Android has a large penetration in the worldwide market, thus developing an application that is able to run on Android devices is very interesting. Combining these arguments leads to the main research question: *When utilising only environmental sound that originates from unknown positions, what techniques for Cooperative Localisation can be used on Android devices that can achieve accuracy within several metres, and what factors will influence this accuracy?*

In order to answer this question this thesis introduces the Cooperative Localisation on Android with ambient Sound Sources (CLASS) Algorithm. This Algorithm produces a location set for all devices and a set of directions towards the origins of the sound-events. A Histogram Based Outlier detection Algorithm is implemented to find outliers in the localisation results. The CLASS Algorithm deals with inaccurate measurement data by finding and averaging TDOA values, and localisation results, that are inliers.

To our best knowledge no prior work has utilised Android for Cooperative Localisation. The following question is therefore posed: *What are the technical limitations of utilising a non Real-Time Operating System like Android for Cooperative Localisation that achieves accuracy within several metres?*

This thesis argues that input latency and poor time synchronisation are the main limitation of the Android Operating System (OS) for its use in Cooperative Localisation by sound. Input latency in Android suffers from large jitters that cannot be predicted and corrected. The following technical limitations of Android, ordered from most to least significant, contribute to TDOA measurement inaccuracies: (i) audio input latency, (ii) poor time synchronisation, (iii) difference in microphone gain per device, (iv) delays in recording time-stamps, (v) implementing a Digital Signal Processor (DSP) like Fast Fourier Transformation (FFT), (vi) noise in the form of peaks in the microphone signal. These limitations can result in erroneous TDOA measurements and are dealt with in the CLASS Algorithm and Android application.

The accuracy of the CLASS Algorithm was assessed with an outdoor experiment. Sound signals were generated with an air horn at twenty locations around a constellation of 16 Nexus-7 tablets. Four sound-events were generated at each location. The sound-events were distributed along a circle with a radius of 46 m . The devices were placed in a 12 $m \times 12 m$ grid with a perpendicular inter-device distance of 4 m .

A mean Root Mean Square Error (RMSE) of 2.4 m with a standard deviation of 0.21 m is achieved. The mean RMSE of the estimated directions is 76.24° with a standard deviation of 1.28°. These results can be improved in future work by elaborating on different parts in the CLASS Algorithm and Android application.

Thesis Supervisor: Paul Havinga

Title: Full Professor

Acknowledgements

I started the preliminary work for my master thesis at the Pervasive Systems group in September 2014. The initial idea was to develop a working demo on Android devices that consisted of a network of devices and could locate themselves by only using sound. This soon proved to be very ambitious to finish within one master thesis. However the work presented in this thesis is definitely a large step in that direction. I want to thank my thesis supervisor, Paul Havinga, for his guidance and input which improved my research skills during the writing of this thesis. In September 2014 Le Viet Duc was working on Android audio input latency at the time and we started to investigate this together. After writing a paper with Le Viet Duc that was related to Android audio latency I started to work on this thesis in January 2015. My thanks goes out to Le Viet Duc for his collaboration and insights, which taught me how to look at research problems with a statistical mindset. I would also like to thank Johannes Wendeberg and Simon Burgess for their feedback and suggestions. Their work really helped me to understand the mathematical problem setting of Cooperative Localisation. My thanks goes out to Okan Turkes for sharing his knowledge and insights of his Cocoon data dissemination model.

Contents

1	Introduction	9
1.1	Goal	10
1.2	Research Questions	10
1.3	Challenges	11
1.4	Related Work	12
1.4.1	Target Localisation Utilising Anchor Devices With Known Positions	12
1.4.2	Cooperative Relative Localisation Utilising Only Sound Signals	13
1.5	Thesis Layout	15
2	Methodology and Approach	17
2.1	General Approach	17
2.2	Assumptions and Requirements	18
2.3	Materials and Equipment	19
2.4	Measurement Set-Up	19
2.5	Measurement Scheme	20
2.6	Dataset and Localisation Results	21
2.7	Conclusion	22
3	Localisation Algorithm	23
3.1	Problem Setting	23
3.1.1	Time Difference of Arrival Matrix	24
3.2	Optimizing for Both Device and Sound-Event Location	25
3.3	Optimizing for Device Location Under the Far Field Approximation	25
3.4	Solver	26
3.5	Averaging TDOA Values for Events at Identical Locations	26
3.6	Outlier Detection in the Results	27
3.6.1	Histogram Based Outlier Detection	29
3.7	Conclusion	31
4	Technical Analysis and Data Acquisition	33
4.1	Related Work	34
4.2	Android Architecture	34
4.3	Time Synchronisation	34
4.3.1	Global Positioning Service	35

4.3.2	Network Time Protocol	36
4.3.3	Measuring Clock Offsets	37
4.4	Android Audio Input Pipeline	40
4.4.1	Input Latency	40
4.4.2	OPEN-SL	42
4.4.3	Audio Input Filter	42
4.4.4	Determining the Desired Frequency Range	43
4.5	Sound-Event Detection	43
4.5.1	Deriving the Desired Threshold Value	44
4.5.2	Averaging Input Samples	44
4.6	Time-Stamping	45
4.7	Networking	45
4.7.1	Server	45
4.8	Data Quality	46
4.8.1	Placing all Devices at an Identical Location	46
4.8.2	Distance Measurement	50
4.9	Conclusion	54
5	Experimental Validation	57
6	Conclusion	61
6.1	Future Work	63
	Glossary	69

Chapter 1

Introduction

A localisation system determines the location of users or objects, either relative to a known position or within a coordinate system. Examples of applications that rely on localisation information include (but are not limited to), law enforcement, military, security, tracking personnel, vehicles, road safety, situation awareness, and mobile ad-hoc networks. This thesis addresses the issue of localising multiple devices in an ad-hoc sensor network; in particular the localisation of devices that have no knowledge of their surroundings but are able to communicate with each other. Techniques that accomplish this are known as Collaborative Localisation, Cooperative Localisation, and Network Localisation [1]. The work presented in this thesis utilises sound signals that are present in the environment. This eliminates the need of existing infrastructure, such as anchor nodes with known locations.

Ideally, each device would know its own position, for example, from the Global Positioning System (GPS). However, constraints such as size, cost, accuracy and limited connectivity require investigation of techniques other than GPS [2].

- Dense forests, mountains, or other obstacles that block the line-of-sight from GPS satellites, will hinder the use of GPS.
- The power consumption of GPS will reduce the battery life of sensor nodes and reduce the effective lifetime of the entire sensor network.
- In a sensor network with large number of devices, the production cost factor of GPS is an important issue.
- Civilian GPS has a horizontal accuracy of approximately four to ten meters, depending on the implemented hardware and environment [3]. Some applications require a higher accuracy.

For these reasons an alternate solution of GPS is required which is cost effective, rapidly deployable and can operate in diverse environments. When geographic positioning, such as GPS, fails in harsh environments, communication between wireless nodes can be used to improve the accuracy of location information [4]. There are numerous methods that have been investigated to preform Cooperative Localisation [2]. One of these methods is based on Time Of Arrival (TOA) measurements. A signal

is transmitted by an anchor node and each device records the time when it receives the respective signal. Knowing the speed of the signal, this difference in time relates directly to the distance between the two devices and the anchor node. This method requires the synchronisation of the devices their clock with the anchor node.

A variation of this method is based on Time Difference Of Arrival (TDOA) measurements. TDOA estimation requires the measurements of the difference in time between the signals arriving at two devices. One device functions as a ‘base’ node and the measured TDOA value, as the name suggests, is the time difference of arrival. The advantage that TDOA has over TOA is that the devices performing the measurements do not have to synchronise their clocks with the anchor node that produces the signal.

In order to eliminate the need of existing infrastructure, such as anchor nodes with known locations, signals already present in the environment should be used. This fact makes the use of sound very interesting. Sound signals are ubiquitous and can be utilised to perform TDOA based Cooperative Localisation. This thesis focusses on research that gears towards ubiquitous Cooperative Localisation using only ambient sound, without the means of any existing infrastructure. This thesis presents the Cooperative Localisation on Android with ambient Sound Sources (CLASS) Algorithm.

1.1 Goal

The goal of this thesis is to develop a Cooperative Localisation method without a dependency on existing infrastructure. This requires a network of devices that can localise their relative positions by using only ambient sound that is present in the environment. In theory this is possible with dedicated hardware [5, 6]. In order to create an application that can easily be used on various hardware platforms it should be able to run on smartphones. Previous work has applied the technique with iPhones and other Apple products [7–9]. Due to its large penetration in the worldwide market, the Android OS is a promising candidate to utilise as a development platform. Android is a leading OS that runs on smartphones, tablets and wearables. Android is not a Real-Time OS and thus using it as a development platform for Cooperative Localisation is challenging. The secondary goal of this thesis is to explore the limitations of implementing the Android OS for Cooperative Localisation.

1.2 Research Questions

The main research question this thesis is trying to answer is:

When utilising only environmental sound that originates from unknown positions, what techniques for Cooperative Localisation can be used on Android devices that can achieve accuracy within several metres, and what factors will influence this accuracy?

Sub-questions that arise from this are:

1. *What technique localises devices in a network without prior knowledge of the device locations and sound signal origins?*
2. *What are the technical limitations of utilising a non Real-Time OS like Android for Cooperative Localisation that achieves accuracy within several metres?*
3. *What are the consequences of inaccurate TDOA measurements for Cooperative Localisation and how can they be minimised to achieve accuracy within several metres?*
4. *What techniques can be used to simultaneously detect an identical sound signal on all devices in a Cooperative Network?*
5. *What type of sound signals can be used for Cooperative Localisation?*
6. *What techniques can be used to share information in a Cooperative Network without using existing infrastructure?*
7. *What kind of information do devices need to share within the network for Cooperative Localisation by sound?*

1.3 Challenges

Localisation by sound introduces multiple challenges such as:

1. Setting up the network for sound localisation
2. Time synchronisation amongst smartphones in the network
3. Detecting and identifying sound-events that can be used for triangulation
4. Audio latency in the hardware platform
5. Jitter in the audio latency
6. Dealing with inaccurate measurements

These challenges and related issues will be discussed in the respective Sections of this thesis. Challenges one to four have been tackled in the experimental Android Application. Challenges five and six have been addressed by the CLASS Algorithm that is introduced in Chapter 3.

1.4 Related Work

Localising an emitter or receiver in a Wireless Sensor Network (WSN) has been studied extensively over recent years [2, 10]. Many have studied the problem of localising a networked device with only sound. In the field of localisation, using only acoustic signals, two approaches can be distinguished; localisation with- and without anchor devices. A lot of research focuses on localising a target node by utilising anchor devices with a known position, this is discussed in Section 1.4.1. Section 1.4.2 describes research where no anchor devices are required. In this field a lot of researchers have tried to overcome measurement inaccuracies by means of ranging between the devices. Ranging is a phase in the localisation Algorithm where devices estimate the respective distances between them and others. More recent work has found solutions that require no ranging between devices, see Section 1.4.2.

1.4.1 Target Localisation Utilising Anchor Devices With Known Positions

Localizing a node when the base stations are known has been studied by many. Harter et. al. introduce a platform that enables applications to follow mobile users and/or objects inside a building with so called ‘Bat’ nodes [11]. After overhearing it’s unique identifier, that is emitted by a base node, a Bat node emits an ultrasound signal that is picked up by an array of receivers mounted in the ceiling. The location of the Bat node is obtained through multilateration with a few centimetres of accuracy. The Bat system requires an extensive network of multiple receivers in the ceiling and base stations that can emit sound signals to the Bats. Every user or object has to carry a Bat node. Although the system is accurate, it requires a lot of infrastructure and is an obtrusive technology to the user.

Simon et. al. introduce PinPtr, an ad-hoc wireless sensor network system that detects and accurately locates shooters [12]. PinPtr consists of a large number of cheap sensors communicating through an ad-hoc wireless network. The dense deployment of sensors detect and measure the TOA of muzzle blasts and shock waves from a gunshot. The sensors route their measurements to a base station that computes the location of the gunshot. PinPtr utilises second-generation Mica2 motes. These motes contain dedicated hardware and the authors were able to synchronise them with an average time synchronization error that stayed below $17.2 \mu s$ during a four hour experiment. The error in time synchronisation between Android devices will be much higher than this [13]. PinPtr requires a ranging phase in order to determine the relative positions of the motes. This is a drawback because the motes require a sounding device that has a limited range of approximately $10 m$.

Peng et. al. introduce an acoustic-based system for ranging and localisation. The authors discuss uncertainties in the measurements of audio localisation and present a high accuracy solution [14]. The proposed method achieves $1 cm$ accuracy for ranging and three cm accuracy for localisation. This can be done by emitting and receiving signals in turns. The devices record their own emitted sound and use the recording to determine their own audio latency. The method does however require anchor devices

and a ranging phase, thus it is not a fully pervasive solution.

Hoflinger et. al. propose a system that utilizes embedded ARM processors as anchor devices [15]. The proposed method localises smartphones that emit short acoustic signals beyond the audible range. In this solution all the anchor devices are stationary, have known positions and a synchronized clock. The authors show that, in an indoor environment, distances less than 10 *m* result in a sound-event detection rate of approximately 80 %. Larger distances resulted in a lot of missing data points. In an outdoor environment there exists a lot of environmental noise such as wind and cars, which are highly likely to increase the error in the sound-event detection rate. Thus, a solution that requires the device to emit a signal outdoors, over larger distances, is not feasible.

Shang et. al. use Android devices as anchors at known locations to estimate the sound source location [13]. The authors find the TDOA by means of cross correlation between sound recordings that were recorded at different known locations. When a recording request has been broadcast through the wireless network, each device starts recording the sound at different locations. All devices will not start recording the sound at exactly the same moment in time. Therefore the start time of the sound recording is stored, so that the start time offset in the TDOA values can be removed later. The authors do not mention how this time stamp is recorded or how accurate these time stamps are. When the recording has finished, each device sends its recorded sound signal, corresponding time stamp and device location to a common device for processing. The TDOA value contains the synchronisation offset and start time error. These two errors are estimated and subtracted from the time delay to obtain the calibrated TDOA for the positioning of the source. The localisation errors of their acoustics-based method are within +/- 15 *cm* in the x-direction and +/- 80 *cm* in the y-direction. The authors conclude that the accuracy in the x-direction is better because the baseline of the sensor array in the y-direction is relatively short (≈ 190 *cm*) compared to the distance from the centroid to the source (≈ 260 *cm*). The baseline in the x-direction is approximately 250 *cm*. The authors observe random error from different trials but do not mention possible reasons for this.

The authors found that multiple independent location estimates can be averaged to yield much better localisation accuracy. The measurements were performed in a quiet environment with low noise levels, outdoors this method has not been tested and could possibly fail. The authors do not mention the inevitable offset in the anchor devices' their location. This offset will probably increase the error in the estimation of the target's location.

1.4.2 Cooperative Relative Localisation Utilising Only Sound Signals

The complexity of the localisation problem increases when base stations with a known position are not present. In this case, there are more unknown parameters that need to be estimated. Most research for this approach also localises the location and time of emission of the sound-event since this information is inherent in the

measurement data. In order to overcome measurement inaccuracies, some authors have implemented ranging.

Utilising Ranging Signals Between Nodes

Raykar et.al preform an analysis of the localisation errors due to imprecise synchronisation and propose ways to account for the unknown speaker emission start times and microphone capture start times [16]. The authors fix the location of some devices in a plane to prevent rotation and translation of the location solution. The proposed method accounts for the lack of time-synchronisation in different platforms by use of ranging.

Hennecke et. al. propose a localisation method that utilizes smartphones [17]. The authors focus on the calibration of low-quality unsynchronized mobile phone audio hardware and resort to acoustic calibration signals to approach the self-localisation task.

The research listed above does show that ranging can be used to compensate for poor time-synchronisation between devices. However, approaches that require ranging are not well suited for outdoor applications, simply because a smartphone will not be able to emit a distinguishable sound-event over larger distances. Therefore it is interesting to look at approaches that require no ranging.

Utilising Only Ambient Sound Without Ranging

Refining the work in [5], Thrun et. al. are one of the first authors to introduce a solution towards localisation without any other information than TDOA measurements of acoustic events over multiple nodes [6]. The authors show that the sensor nodes localisation problem is equivalent to a Maximum Likelihood Estimation (MLE). In their approach the authors rely on the Far Field Approximation (FFA). When there is no ranging between nodes, most prior works assume that acoustic events originate (infinitely) far away so that sound waves arrive at all devices in parallel. This assumption simplifies the problem statement as discussed in Section 3.3.

The FFA is refined by Kuang et. al. in [18]. Their experimental validation gives a strong indication that a FFA is a feasible approach for getting direct estimates as well as initial estimates for other solvers.

Wendeberg et. al.[7, 8] have successfully localised a group of networking devices in a mobile environment without the need of any further infrastructure besides ambient sound and a Wi-Fi network. They utilised the TDOA method to localise a network of Apple laptops and iPhones with a positioning accuracy of 10 *cm*. The Apple products they used contain a High Precision Event Timer (HPET), thus it becomes possible to synchronise the devices quite accurately. Wendeberg et. al. were able to synchronise the device's their system clocks within an accuracy of 0.1 *ms*. In Android this accuracy in time synchronisation is not possible and the measurements will be more noisy.

In order to deal with noisy data, Burgess et. al. constructed an Algorithm that uses the Random Sample Consensus (RANSAC) paradigm [9]. Their method simulta-

neously solves the calibration problem and removes severe outliers, which is a common problem in TOA applications. With two indoor environment experiments, using microphones and speakers, their work achieved a Root Mean Square Error (RMSE) of 2.35 *cm* and 3.95 *cm* on receivers and transmitters their respective positions compared to computer vision reconstructions.

To our best knowledge there is no prior work that has tried to develop a Cooperative Localisation Algorithm on Android Devices. Most likely Android's indeterministic behaviour has made the OS an unpopular platform to use for Cooperative Localisation by sound. Investigating Android's applicability for Cooperative Localisation is one of the contributions in this thesis.

1.5 Thesis Layout

The thesis is organised as follows. Chapter 2 describes the methodology of the conducted research. Chapter 3 lays out the problem setting and describes the CLASS Algorithm. Chapter 4 discusses the details around the acquisition of the data-set that was used with the CLASS Algorithm, and investigates the technical limitations of the Android OS. Chapter 5 validates the CLASS Algorithm by discussing the experimental results. Finally the thesis is concluded along with the future work in Chapter 6.

Chapter 2

Methodology and Approach

This Chapter describes the methodology of the conducted research.

Section 2.1 introduces the general approach. Section 2.2 discusses the research assumptions and requirements. The materials and equipment that have been used during the research are described in Section 2.3. The measurement set-up of the experimental application is discussed in Section 2.4. Section 2.5 describes how the outdoor experiment was conducted. Section 2.6 describes the conditions that were applied to the raw data-set in order to assemble the final data-set.

2.1 General Approach

A network of mobile devices was required to answer the research questions posed in this thesis. Because the second objective was to investigate the limitations of utilising the Android OS, Android operated devices were used. A cooperative network was created with these devices.

After reviewing the mathematical problem setting, the CLASS Algorithm was developed that utilised data, acquired during experiments described in Section 2.5, to locate the position of the devices. The CLASS Algorithm consists of filtering erroneous TDOA measurements, averaging inliers, and finding an optimal location for all devices and an angular directions towards the origin of the sound-event locations. The CLASS Algorithm is described in Chapter 3.

An experimental Android application was developed and analysed. The experimental Application and Android limitations are discussed in Chapter 4. The application networked the devices and allowed them to respond to sound-events, originating from outside the device constellation.

In order to obtain a realistic data set, the main experiments took place in an outdoor environment, where the set-up was exposed to wind and environmental noise. The sound-events were generated by means of an air horn which generated a loud, distinctive sound with a certain frequency. In general any type of sound can be used when it is loud enough and can be distinguished by all devices.

For simplicity reasons the network was configured as a star-network. The devices were wirelessly connected through a local Wi-Fi network and all measurement data

was transferred to a database located on a Laptop through the Hypertext Transfer Protocol (HTTP). The CLASS Algorithm was later executed offline.

2.2 Assumptions and Requirements

The requirements for the localisation application presented in this thesis are as follows:

1. In order to reconstruct the relative positions of the devices, without prior knowledge, sound-events must originate from different locations, outside and around the device constellation. Without this requirement there will be insufficient information regarding the relative locations of the devices.
2. Sound-events must be spaced sufficiently far apart in time to distinguish individual events.
3. Sound-events should have a significant Signal to Noise Ratio (SNR) so that they can be overheard by all devices in the constellation.
4. In the experimental application audio input samples are averaged in order to filter noise, sound-events must therefore have a duration, above the SNR, of at least $200 \mu s$.
5. The sound-event does not have to be constant, but in order to minimise the error in TDOA values, sound-events should have a short onset time. In other words, the signal must have a steep flank e.g. clapping, gun shots, or an air horn.
6. The CLASS Algorithm, described in Chapter 3, requires a minimum of six sound-event locations and three devices to find a solution [9].
7. The method described in this thesis relies on the FFA. The FFA is discussed in Section 3.3. Sound-events have to originate at a minimum distance from the device constellation for the FFA to hold. Thrun S. found that the localisation error did not change significantly over distances larger 5 times the diameter of the sensor array [6]. This distance is the diameter of the sound-event locations when they are distributed in a circle around the device constellation. With the set-up described in Section 2.4, the maximum device constellation diameter $17 m$. The diameter of the sound-event locations should therefore be equal to $5 \cdot 17 = 85 m$. The minimum distance from the edge of the device constellation then becomes $\frac{85-17}{2} = 34 m$.
8. Devices should not be placed too far apart from each other. When the inter-device distance becomes too large, some devices might overhear a sound-event, whilst the sound signal fades out for devices that are far away.
9. The approach presented in this thesis requires the use of mobile devices that can run the Android OS. The requirements for these devices are as follows:

- (a) A device must have the capability to record sound.
 - (b) The experimental application requires a Wi-Fi and GPS radio for data transmission and time synchronisation.
 - (c) A device that acts as a ‘server’, which is used to start a sound-event measurement, must have hotspot mode availability.
 - (d) The devices have to be synchronised in time. To estimate the TDOA it is important that the time difference between the system clocks of the devices is minimal.
10. The RMSE of the location solution for the devices must be smaller than 6.32 *m*. The simplest Algorithm would be to locate all devices on top of each other. According to Equation 2.1, placing all devices in the centre of the set-up described in Section 2.4, results in a RMSE of 6.23 *m*. Thus any solution with a RMSE larger than 6.23 *m* does not make sense.

2.3 Materials and Equipment

Sixteen Nexus-7 (2012 edition), tablets have been used to obtain the data that is used in this research. All devices were updated to Android version 4.4.4 (KitKat).

A Lenovo W540 laptop has been used to store the data during the measurements, and later to execute the CLASS Algorithm.

All the data was transferred from the devices via a local Wi-Fi network. A TP-LINK (Archer C7 - Wireless AC1750) was used to set up the network. During measurements in the field a 12 V battery was used to power the router.

The sound-events were generated with an air horn. More details regarding the air horn are discussed in Section 4.4.4.

2.4 Measurement Set-Up

Figure 2-1 displays a top-down 2D overview of the ground truth for device locations *S* and sound-event origins *E*. Sound-events were generated from twenty different locations distributed along a circle around the device constellation. To make sure the FFA was not violated, the sound-events were generated forty metres away from the edge of the device constellation.

Sixteen devices were placed on top of tripods, at a height of approximately one metre, in a 12 *m* × 12 *m* grid with a perpendicular inter-device distance of four metre. Sixteen devices were used because of their availability. A minimum of three devices or more than sixteen devices can be used. Using more devices increases the computational complexity. At some point the Cooperative Localisation would need to be split up, e.g. by dividing the network and localising devices in clusters. Devices should not be too far apart from one and another, see Requirement number 8. When the devices are placed too close to each other, the estimation error of the relative device positions becomes larger than the internode distance, which renders

the solution useless. In order to satisfy both Requirements 7 and 8, whilst maximizing the inter-node distance, an inter-node distance of 4 *m* was considered to be optimal.

The devices were placed on top of the tripods with the microphone holes facing in random directions. A photo of the outdoor experiment is shown in Figure 2-2.

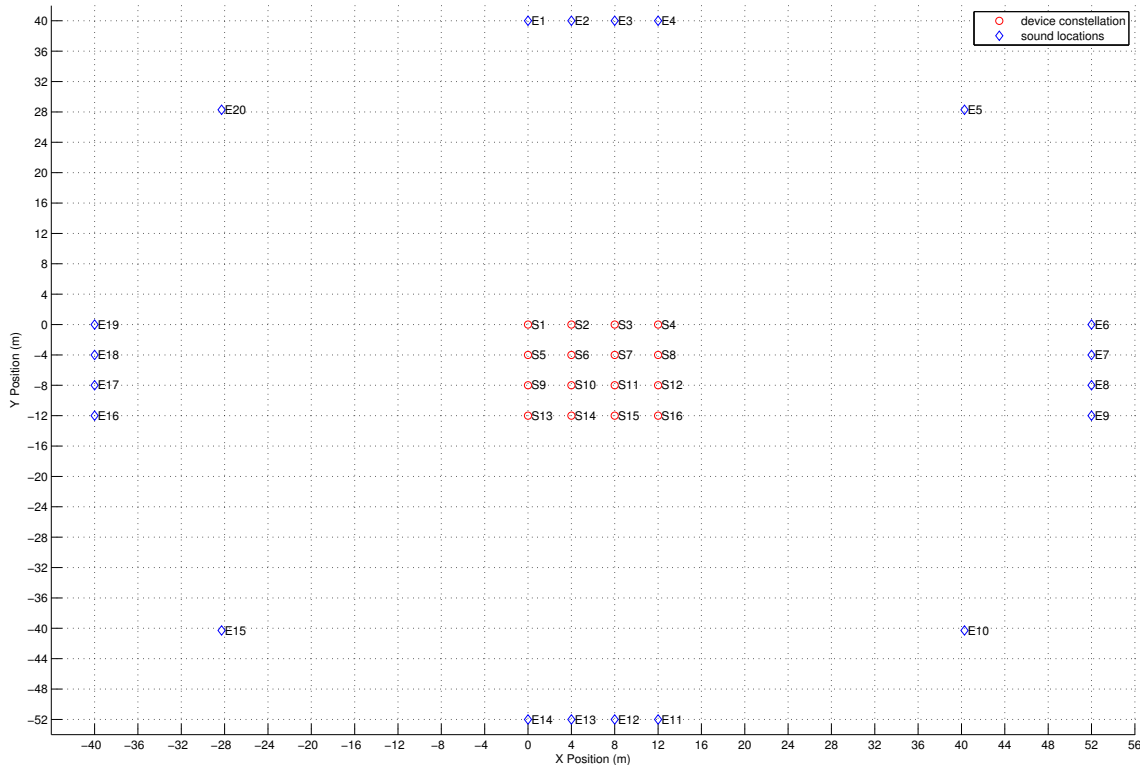


Figure 2-1: Experimental Outdoor Set-up

2.5 Measurement Scheme

The following steps were taken during the experimental measurements. A remote device running the experimental application took the role of ‘server’ and all other devices acted as a ‘client’. In the experiment the sound-event was known to be generated with the air-horn. The Android application on each device was configured with a band pass filter that filtered out environmental sounds and let the frequency band of the air horn pass through. The devices triggered when the SNR of the air horn exceeded a pre-set threshold value. This is discussed with more detail in Section 4.5. The SNR threshold, together with the location of the sound-event, were configured on the server device before each measurement. When a new measurement was initiated, the server device broadcast a start message, along with a measurement id, the SNR threshold and sound-event location. All devices started with a ‘calibration’ phase. During this phase all devices determine the noise level of the environment for three seconds. After this phase the devices started to listen in the frequency band of the



Figure 2-2: Experimental Outdoor Set-up

air horn's signal by filtering out all other frequencies. As soon as the energy level in this band reached the threshold, a time-stamp was recorded on each device. A sound-event was generated by sounding the horn in the direction of the device constellation. The devices then recorded and sent a time-stamp along with the measurement-id and sound-event location to the server. The server stored all the values in a Structured Query Language (SQL) database. This process was repeated multiple times per sound-event location.

2.6 Dataset and Localisation Results

The following conditions were applied to the raw data to distinguish a successful measurement.

1. All devices must have triggered on the sound-event. In practise the CLASS Algorithm can deal with missing data points, however to asses the quality of the data it is required to only include measurements where all devices detected the sound-event.
2. The standard deviation over the TOA values of all devices has to be smaller or equal to 30 ms . By the speed of the sound, 30 ms relates to a standard deviation of approximately 10 m . A recorded TOA set that has a standard deviation that is larger than 10 m is considered to contain too many outliers. If for some reason one or more devices triggered significantly too early or too late, they caused a high standard deviation in the TOA set for that sound-event. These reasons include (but are not limited to); sounding the horn too early, sounding the horn with insufficient volume, and a faulty calibration phase which resulted a SNR threshold that was above the energy of the air horn signal. This occurred occasionally during the measurements and these data were not included in the dataset.

The CLASS Algorithm returns a solution set for the location of all devices \bar{X} and a set of directions $\bar{\alpha}$ towards all sound-event origins, see Figures 5-1 and 5-2 in Chapter 5, respectively. Each direction α_j is denoted as a vector that originates at the centre of the device constellation and is oriented towards sound-event E_j . The obtained solution for the position of all devices is always a relative position to other nodes, there is no known absolute position. Localisation without an anchor device results in a solution set that is rotated and translated. In order to evaluate the quality of the solution, it is matched to the ground truth by rotation and translation. The solution is rotated and translated to best alignment with the ground truth. The rotation and translation that results in the optimal fit are found by using [19]. The relative error of the solution is then calculated as depicted in Equation 2.1.

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N \left(\|X_i - \bar{X}_i\|_2 \right)^2} \quad (2.1)$$

Where N denotes the number of devices, $\|\cdot\|_2$ the L^2 norm (Euclidean distance), X_i the ground truth location of device i and \bar{X}_i the estimated location.

In order to assess the quality of the estimated set of directions $\bar{\alpha}$ is rotated to best fit the ground truth. The relative error for the estimation of the directions is then calculated as depicted in Equation 2.2.

$$RMSE = \sqrt{\frac{1}{M} \sum_{j=1}^M (\alpha_j - \bar{\alpha}_j)^2} \quad (2.2)$$

Where M denotes the number of sound-events, α_j the ground truth and $\bar{\alpha}_j$ the estimated direction towards sound-event j .

2.7 Conclusion

This Chapter systematically solved the research problem by identifying methods, techniques and metrics for evaluation. Chapter 3 continues with identifying the mathematical problem setting for Cooperative Localisation by ambient sound and introduces the CLASS Algorithm.

Chapter 3

Localisation Algorithm

How do devices in a Cooperative Network localise themselves without prior knowledge of the device locations and sound signal origins? How can the consequences of inaccurate TDOA measurements be minimised so that accuracy within several metres is achieved? This Chapter attempts to answer these questions by introducing the mathematical problem setting and describing the CLASS Algorithm that is used to localise the mobile devices based on ambient sound-events at unknown locations.

Section 3.1 starts with a mathematical approach to the problem of Cooperative Localisation without any prior knowledge. A cost function for the localisation of all nodes and sound-event origins is formulated in Section 3.2. Section 3.3 elaborates on the cost function and describes a simplified function where an angular direction toward the sound-event origins is used. Section 3.4 then discusses the solver that is used to optimize this cost function to an optimal localisation solution. Section 3.5 describes how multiple sound-events recorded at an identical location can be utilised to optimise TDOA values. Section 3.6 discusses the filtering of outliers in the localisation results by a Histogram Based Outlier Score (HBOS) Algorithm. The Chapter is concluded in Section 3.7.

3.1 Problem Setting

This Section denotes the mathematical formulation of the localisation problem. For ease of representation the sensors and sound-sources are located in a 2D plane. The technique can easily be generalised to use 3D positions. In this problem formulation it is assumed that the detected sound waves travel from the source to the receiver in a straight line.

Let there be N nodes at unknown locations $X \in \mathbb{R}^2$.

$$X = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_N & y_N \end{pmatrix} \quad (3.1)$$

Let there be M sound events at unknown locations $A \in \mathbb{R}^2$. The times at which

sound-events are emitted are denoted by a vector E . Each sound-event has an unknown emission time e_j .

$$A = \begin{pmatrix} a_1 & b_1 \\ b_2 & b_2 \\ \vdots & \vdots \\ b_M & b_M \end{pmatrix} \quad (3.2) \quad E = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_M \end{pmatrix} \quad (3.3)$$

The time of arrival of each sound-event, at each device is represented by a $N \times M$ TOA matrix T . In this matrix the arrival time of acoustic event j at device i is denoted as $t_{i,j}$.

$$T = \begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,M} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ t_{N,1} & t_{N,2} & \cdots & t_{N,M} \end{pmatrix} \quad (3.4)$$

It is assumed that each event is well distinguished so that there is no data association problem. Each sound-event represents one column in T_{ij} .

t_{ij} can be denoted as the sum of its respective emission time e_j and the time it takes to travel from position (a_j, b_j) to (x_i, y_i) . This relation can be denoted as:

$$t_{i,j} = e_j + \frac{1}{c} \cdot \left\| \begin{pmatrix} x_i \\ y_i \end{pmatrix} - \begin{pmatrix} a_j \\ b_j \end{pmatrix} \right\|_2 \quad (3.5)$$

Where c denotes the speed of sound and $\|\cdot\|_2$ denotes the L^2 norm (Euclidean distance).

3.1.1 Time Difference of Arrival Matrix

When no absolute positions are known we can only look at the relative positions of the nodes. Therefore we look at the inter-device Time Difference Of Arrival (TDOA). E.g. the TDOA between device 1 and all other devices can be denoted as:

$$\Delta = \begin{pmatrix} t_{2,1} - t_{1,1} & t_{2,2} - t_{1,2} & \cdots & t_{2,M} - t_{1,M} \\ t_{3,1} - t_{1,1} & t_{3,2} - t_{1,2} & \cdots & t_{3,M} - t_{1,M} \\ \vdots & \vdots & \ddots & \vdots \\ t_{N,1} - t_{1,1} & t_{N,2} - t_{1,2} & \cdots & t_{N,M} - t_{1,M} \end{pmatrix} \quad (3.6)$$

When we define the TDOA values not only relative to node 1, but between all pair of nodes, we obtain the matrix Δ with dimensions $(N-1) \times M \times N$. In this matrix the TDOA between node i and k for sound-event j is denoted as Δ_{ijk} .

Multiplying a TDOA value with the speed of sound yields the euclidean distance between the two respective nodes. The relation between the TDOA and euclidean distance can be denoted as:

$$\begin{aligned}\Delta_{ijk} \cdot c &= \|X_i - A_j\|_2 - \|X_k - A_j\|_2 \\ 0 &= \|X_i - A_j\|_2 - \|X_k - A_j\|_2 - \Delta_{ijk} \cdot c\end{aligned}\tag{3.7}$$

This relation can be used to form a function that can be minimized in order to find an optimal solution for device locations X and sound source locations A . This cost function is described in Section 3.2.

3.2 Optimizing for Both Device and Sound-Event Location

From Equation 3.7 we can form a least squares definition as denoted in Equation 3.8.

$$J(X, A) = \sum_{i=1}^{N-1} \sum_{j=1}^M \sum_{k=i+1}^N \left(\|X_i - A_j\|_2 - \|X_k - A_j\|_2 - \Delta_{ijk} \cdot c \right)^2\tag{3.8}$$

Where X_i denotes the device's location (x_i, y_i) and A_i denotes the sound-event's location (a_j, b_j) . A common approach is to optimize the parameters A and X using an iterative Algorithm such as the Levenberg-Marquardt [20] or Trust-Region-Reflective Algorithm [21]. The iterative Algorithm returns a set of parameters for which the residual J of function 3.8 is minimal. The returned set of locations are then considered to be the most optimal solution of the localisation problem. The solver is discussed in more detail in Section 3.4.

3.3 Optimizing for Device Location Under the Far Field Approximation

In [22] Burgess et al. extend on previous work and introduce two novel algorithms for parameter estimation of a receiver array. These algorithms are based on the assumption that sound events originate from (infinitely) far away. Under this assumption a sound wave from sound-event j arrives at each device at the same incident angle α_j . In other words, the sound waves connecting the location of a sound-event (a_j, b_j) with each of the devices (x_i, y_i) are approximately parallel for all i (but not for all j) [6, 22]. This assumption can be used to simplify the localisation problem [18]. Now the solver does not need to find an exact location for each sound-event but merely one direction towards it.

The relation between the inter-device distance, sound-event location and the respective TDOA value, as described in Equation 3.7, can be rewritten as a function of the incident angle α_j as depicted in Equation 3.9.

$$\begin{pmatrix} \cos(\alpha_j) & \sin(\alpha_j) \end{pmatrix} \cdot \begin{pmatrix} x_i - x_k \\ y_i - y_k \end{pmatrix} = \Delta_{ijk} \cdot c\tag{3.9}$$

This leads to the cost function described in Equation 3.10.

$$J(X, \vec{\alpha}) = \sum_{i=1}^{N-1} \sum_{j=1}^M \sum_{k=i+1}^N \left((\cos(\alpha_j) \quad \sin(\alpha_j)) \cdot \begin{pmatrix} x_i - x_k \\ y_i - y_k \end{pmatrix} - \Delta_{ijk} \cdot c \right)^2 \quad (3.10)$$

3.4 Solver

The problem described in Section 3.3 is a non-linear least squares problem. These kind of problems can be solved with algorithms such as the Levenberg-Marquardt Algorithm [20]. This Algorithm is used extensively in various software applications and has proven to be fast and stable.

The Algorithm updates the parameters that need to be estimated with steps towards an optimum where the residual of the cost function are minimal. This Algorithm can take large steps when it is far away from the optimal solution, but will slow down as it approaches the optimum to prevent unstable behaviour. The Algorithm requires an initial guess or 'starting point' to start searching for an optimal solution. The Algorithm updates each parameter so that the residual of the cost function in the next iteration is less than in the current iteration. The Algorithm will always find a local optimum as it only converges in a 'downward' direction from it's starting point. The initial starting point is therefore important in order to find the global minimum. The global minimum is where the residuals of Equation 3.10 are minimal. The estimated parameters in the optimum solution have the best fit to the measured TDOA values.

The parameters that need to be initialized are the angles $\vec{\alpha}$ towards each sound-event from the centre of the device constellation and the location of the devices X . The solver that is introduced in [22] implements a fast Algorithm that aims to solve the same localisation problem as described in this thesis. While it is fast it does not return the most optimal solution for this case, however the result is good enough to use as an initial guess for the Levenberg-Marquardt Algorithm. It is thus implemented in the experimental application to obtain the initial parameter estimation.

3.5 Averaging TDOA Values for Events at Identical Locations

When it is possible to obtain multiple samples from one location by e.g. splitting longer sounds over multiple events, a higher accuracy can be achieved [13]. The recorded TDOA values for an identical sound-location are normally distributed, see Section 4.8. We can use the mean of TDOA values to determine which measurements are inliers.

In the experimental data set four sound-events have been recorded at each sound-event location. This opens up the possibility to use the averaged value of inliers for

the respective value of Δ_{ijk} . Outliers are extracted by means of a HBOS Algorithm. HBOS is explained in more detail in Section 3.6.1.

A value Δ_{ijk} is perceived as an inlier when the condition in Equation 3.11 holds.

$$T_{ik} \cdot c < \epsilon \quad (3.11)$$

Where T_{ik} is the time difference of arrival between device i and k , c is the speed of sound and ϵ the maximum distance from the mean TDOA value ω , which is determined by the HBOS Algorithm. The resulting set with inliers is averaged to form a new value Δ_{ijk} . When no inliers are found, Δ_{ijk} is set to 0. By doing so, a faulty/noisy measurement between device i and k at this location will not take part in the localisation Algorithm. Both devices still have relations with other devices that are more stable and these can be used to determine their respective locations. Thus, removing a TDOA value between a pair of nodes for one sound-event location is not a problem. The method that filters and averages the TDOA values is presented as Algorithm 1.

Although the HBOS Algorithm performs well in this case, it is not the most optimal method to use with very little data points. In this case 4 sound-events were recorded per location which is statistically insufficient to accurately determine outliers. It was implemented with the expectation of having at least 10 data points per sound-location. Due to a limited amount of time it was not possible to obtain this amount of sound-events per location. In future work at least 10 data points should be recorded per location or an alternative method should be investigated for averaging of the TDOA values at identical locations.

3.6 Outlier Detection in the Results

In recent work Burgess et. al. [9] have demonstrated that implementing a RANSAC based scheme is a good method to filter outliers from the dataset. Some input measurements have large errors and will result in an incorrect solution. For each sound-event some devices will not trigger, or will have a large temporal error in the TOA value.

The CLASS Algorithm uses a subset of n devices over multiple iterations. A new random subset is selected per iteration and the results for that subset of nodes is stored. The amount of iterations k is dependent on the size of the subset, see Equation 3.12

$$k = \text{ROUND}\left(\frac{N}{n} \cdot 35\right) \quad (3.12)$$

Where N is the total number of devices in the device constellation and n the number of devices in the subset per iteration. A small subset will have a higher variation in the resulting locations, and thus needs more iterations to determine the inliers. A larger subset will vary less, and needs fewer iterations. The multiplication with 35 has been obtained experimentally. When $n = N$, 35 is the minimum number of iterations to enable robust inlier detection. The optimal subset was determined experimentally by trying different sizes and comparing the results, these are plotted

Algorithm 1: TDOA filtering and averaging

Data: TOA matrix T , maximum inlier distance ϵ , nr events per location \vec{e}
Result: TDOA matrix Δ

```
/* Calculate all TDOA values for all measurements */
1 for every device  $i$  do
2   for every measurement  $g$  do
3     for every device  $k = i + 1$  do
4        $\delta_{igk} = T_{ig} - T_{kg}$ 

/* Determine average TDOA value between all pairs of devices  $i$  and
    $k$  per location  $j$  */
5 for all nodes  $i - 1$  do
6   for all sound-locations  $j$  do
7      $\delta'$  = all TDOA measurements  $\delta$ , between all nodes, that were recorded
      at location  $j$ 
8     for all devices  $k = i + 1$  do
9       divide all TDOA values between device  $i$  and  $k$  in to 3 bins
10       $\omega$  = mean of the bin centre values that have the maximum number
      of counts
11       $\Delta_{ijk} = \frac{1}{n} \cdot \sum_{m=1}^n \delta'_m \Big|_{abs(\delta'_m - \omega) \cdot c < \epsilon}$ 
12      if no inliers found in  $\delta'$  then
13         $\Delta_{ijk} = 0$ 
```

in Figure 3-1. The results indicate that between six and twelve devices, the number of devices does not influence the accuracy significantly.

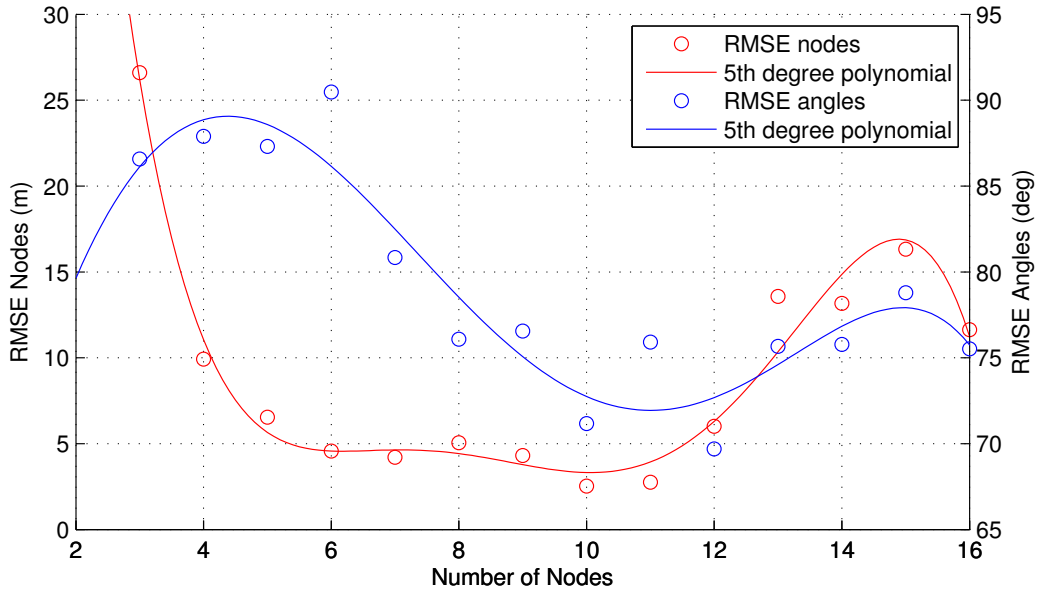


Figure 3-1: Varying the size of the subset of a constellation with 16 devices

The resulting set of location data for each device contains outliers. These outliers are detected and removed by a HBOS Algorithm, see Section 3.6.1.

3.6.1 Histogram Based Outlier Detection

A HBOS Algorithm is implemented to detect outliers in the locations (x, y) of all devices. The Algorithm is presented in pseudo code as Algorithm 2.

In order to determine which values in the set are inliers, a mean, upper and lower bound are first determined. The HBOS Algorithm categorizes all values within a certain range into bins. This is done by creating a histogram of the vector \vec{g} for both x and y values and has size $1 \times k$. The bin-width h of the histogram is determined with the Freedman-Diaconis rule [23], see Equation 3.13.

$$h = 2 \cdot IQR(\vec{g}) \cdot k^{\frac{1}{3}} \quad (3.13)$$

Where $IQR(\vec{g})$ is the interquartile range of the location data and k equals the number of observations in the set \vec{g} .

The number of bins b is based on the size of the bins and the minimum and maximum value in \vec{g} , see Equation 3.14.

$$b = ROUND\left(\frac{MAX(\vec{g}) - MIN(\vec{g})}{h}\right) \quad (3.14)$$

The minimum number of bins b is set to 15, fewer bins will not provide enough resolution for outlier detection. \vec{g} can contain extremely severe outliers which influence the determination of b negatively, \vec{g} is therefore limited to:

$$\vec{g} = \forall m \in \{g_1, g_2, \dots, g_k\} \begin{cases} -1500 < g_m < 0 \\ 0 > g_m > 1500 \end{cases} \quad (3.15)$$

Figure 3-2 displays the bar-plot from a histogram of \vec{g} where the amount of iterations $k = 35$. In this example most of the values in \vec{g} are categorized in bin 3, the centre value that belongs to this bin is used as the mean value ω for inlier determination.

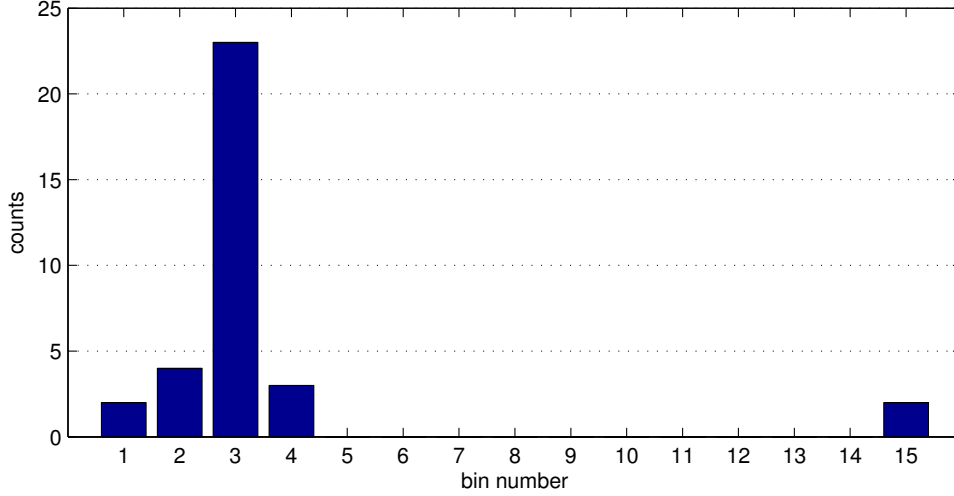


Figure 3-2: Bin classification and bin counts

The bounds are determined by adding, or subtracting a threshold value ϵ . The value of ϵ should not be too small otherwise there will be too less inliers to determine an accurate mean value. When ϵ is too large, false inliers will create an error in the result. The value for ϵ has been determined experimentally, see Figure 3-3

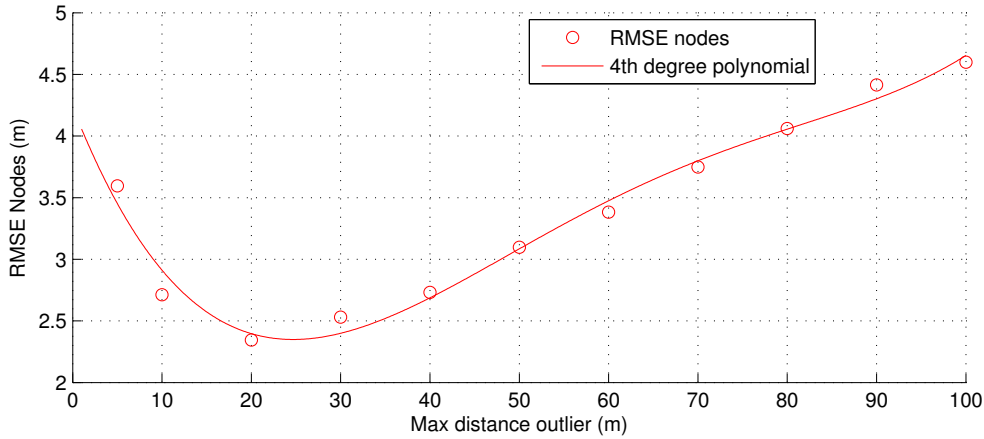


Figure 3-3: Varying the size of maximum distance ϵ for inliers, with 16 devices and subset size of 10 nodes

The set \vec{g} is shown in Figure 3-4 as a scatter plot. The mean value ω and the inlier bounds ϵ are graphed in blue and red, respectively. All values in the area between the red lines are perceived as inliers.

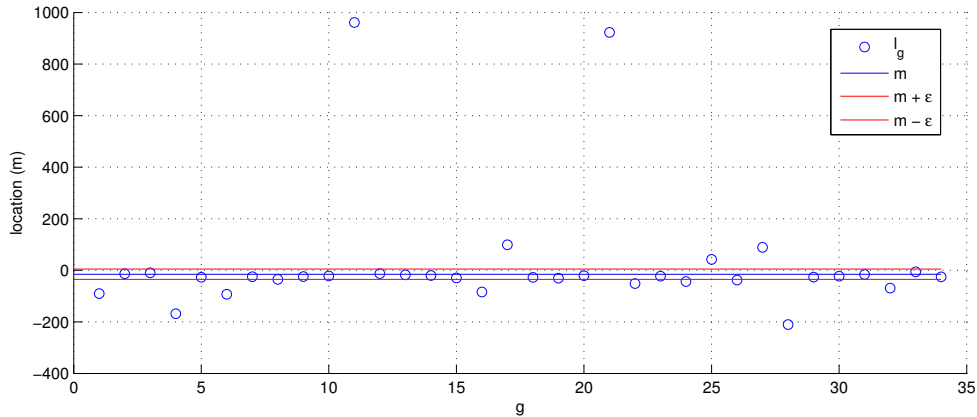


Figure 3-4: Values of x for one device, with mean ω and bounds ϵ . $\epsilon = 20$

It is possible that two bins receive a similar amount of counts, this means that \vec{g} contains arbitrary data. The Algorithm therefore uses the mean value of the centres that have either the maximum number of counts, or are within 10 % of the maximum count. By doing so the most optimal value for ω is found.

3.7 Conclusion

This Chapter explained how devices can cooperate to obtain a relative localisation solution. The relation between the position of devices X , the directions towards the origins of the sound-event locations α and the TDOA values was described. TDOA values that are inliers are averaged for sound-events originating from an identical location in a matrix Δ . The Levenberg-Marquardt solver is utilised to find an optimal solution set for the all the positions and directions from Δ . Finally a HBOS Algorithm is implemented to find outliers in the localisation results. Between six and twelve devices, the subset-size does not influence the accuracy of the localisation solution significantly. The Localisation Algorithm deals with inaccurate measurement data by finding and averaging TDOA values, and localisation results, that are inliers.

Having identified the mathematical problem setting and investigated techniques to deal with inaccurate measurement data, we can now investigate how to acquire the TOA data that is used as input for the CLASS Algorithm. The technical analysis of the Android OS and data acquisition are discussed in Chapter 4.

Algorithm 2: HBOS filter

Data: solution set \widehat{X} ($N \times 2 \times k$), maximum inlier distance ϵ , number of solutions k

Result: estimated location solution \bar{X}

/* Calculate a mean value for x_i and y_i with only inliers for all devices i */

- 1 **for** each device i **do**
- 2 extract all values for x_i from \widehat{X} into g
- 3 $\vec{g} = \forall m \in \{g_1, g_2, \dots, g_k\} \left| \begin{array}{l} -1500 < g_m < 0 \\ 0 > g_m > 1500 \end{array} \right.$
- 4 calculate the bin width $h = 2 \cdot IQR(\vec{l}_i) \cdot k^{\frac{1}{3}}$
- 5 calculate the number of bins $b_i = ROUND\left(\frac{MAX(\vec{g}) - MIN(\vec{g})}{h}\right)$
- 6 **if** $b < 15$ **then**
- 7 $b = 15$
- 8 categorize \vec{g} into b bins
- 9 get the bin centre value of the bins with maximum number count (β_1) and second maximum number of counts (β_2)
- 10 **if** $\frac{\beta_1}{\beta_2} > 0.9$ **then**
- 11 $\omega =$ the mean centre value of the bins with the top 2 maximum number of counts
- 12 **else**
- 13 $\omega =$ the mean of the bin centre values that have the maximum number of counts
- 14 $\bar{x}_i = \frac{1}{n} \cdot \sum_{m=1}^n g_m \left|_{abs(g_m - \omega) < \epsilon}$
- /* Steps 2 - 14 are performed for the values of y_i in an identical fashion */
- 15 $\bar{X}_i = (\bar{x}_i \ \bar{y}_i)$

Chapter 4

Technical Analysis and Data Acquisition

An experimental application has been developed during this thesis in order to obtain a data set with real data. The Android OS is chosen as the application platform because it is a widely distributed operating system and has a large penetration in the market. With 80 %, Android's market share is significantly larger than other mobile Operating Systems such as Apple's iOS [24]. Android is deployed on devices that are distributed by a lot of different vendors which implement different hardware platforms. This generality introduces inaccuracies that can be problematic for high-accuracy demanding applications, such as localisation by sound.

This Chapter attempts to localise the sources of these inaccuracies and discusses the acquisition of sound-event data in a network of mobile Android devices. The acquired sound-event data is used as input for the CLASS Algorithm that was introduced in Chapter 3. Several technical limitations of utilising a non Real-Time OS like Android for Cooperative Localisation will be discussed. Limitations that will be identified are: (i) poor time synchronisation (ii) audio input latency, (iii) implementing a DSP like FFT, (iv) difference in microphone gain per device, (v) noise in the form of peaks in the microphone signal, (vi) delays that occur during the recording of time-stamps.

The Chapter then describes how multiple devices can simultaneously detect an identical sound signal and investigates what kind of information devices need to share within the network for Cooperative Localisation by sound.

Work related to developing Real-Time applications on Android is reviewed in Section 4.1. Section 4.2 describes the overall Android architecture. Section 4.3 describes two methods that have been considered to synchronize a network of Android devices in time. Section 4.4 describes the audio input pipeline requirements. Section 4.5 discusses how a sound-event is detected in the environment. Section 4.7 discusses the network requirements and solutions for a sound localisation application. The quality of the acquired TDOA data is assessed in Section 4.8. Finally the Chapter is concluded in Section 4.9.

4.1 Related Work

In order to use Android as a platform that collects accurate data, the OS requires Real-Time (RT) specifications. There has been recent interest in exploring the addition of RT features to Android [25–30]. In [31] the authors present RTDroid, a variant of Android that provides predictability to Android applications. They replace the standard Dalvik Virtual Machine with a RT Virtual Java Machine and, for hard real-time behaviour, replace the Linux kernel with a certified RT Operating System (RTOS) such as RTEMS [32]. They show that just replacing these elements is insufficient to run an Android application with RT guarantees. After redesigning Android’s core constructs and system services, they were able to provide tight latency bounds to RT applications. In [33] the same authors extend on their previous work and measure it against JPapaBench, a RT Java benchmark. In this work they examine the Android’s sensor architecture in detail and show why it is not suitable for use in a RT context. They then introduce a re-design of the sensor architecture and show that the re-designed sensing architecture can provide predictable performance. Their work also shows the amount of effort that is needed to transform Android into a platform with RT specifications. No modifications were made to the Android architecture for the application described in this thesis.

4.2 Android Architecture

Android is run on devices from many different manufactures. Therefore the OS has to be universal. In order to achieve this universality the Android architecture is abstracted in to several layers, see Figure 4-1. High level applications can easily implement many classes from the Android framework without having to deal with low level systems, such as device drivers etc. The downside of this abstraction, as will be discussed in later sections, is the amount of overhead between an user application in the top layer and low-level systems, such as the audio hardware. Android is not a Real-Time OS which makes it challenging to use for a sound localisation application.

4.3 Time Synchronisation

In order to obtain the location of devices and sound-events, each participating device must record a time stamp as soon as it detects a distinct sound-event in the environment. In order to minimize errors in the time stamps, it is mandatory that the system clocks between the devices are synchronized. An error of 10 *ms* in the time stamp relates to a spatial error of 3.40 *m*, an accurate time synchronisation smaller than 1 *ms* is therefore desirable.

In this thesis two methods for time synchronisation between nodes in a network have been considered; using GPS time information and synchronisation through Network Time Protocol (NTP). Both methods are discussed in sections 4.3.1 and 4.3.2 respectively.

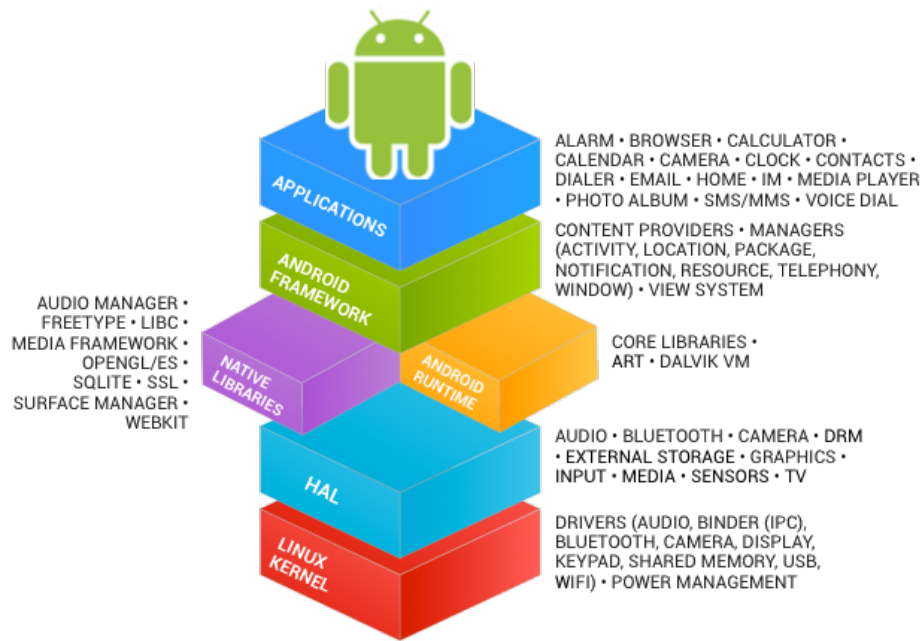


Figure 4-1: Android stack. Source: <https://source.android.com/source/index.html>

4.3.1 Global Positioning Service

All devices in the experimental network are equipped with a GPS chip. The GPS architecture that is used in the Android OS contains the following components:

GPS Chip The physical Radio Frequency Receiver that communicates directly with GPS Satellites

GPS Driver GPS Driver System Software that uses Low level Application Programming Interface (API)s to Communicate with the GPS chip

GL Engine The main component in the overall GPS architecture. The GL engine uses configuration parameters combined with stored data in memory and initial location data from cell towers to instruct the GPS driver

Android Framework Location and GPS satellite data is communicated to the Android Framework using the GL engine.

User Applications User applications can implement the Android Location Service classes to use GPS data

When a GPS radio is connected to multiple satellites the device can acquire very accurate time synchronisation with the satellites. This accurate time ρ is communicated through so called NMEA messages and has a resolution of nano seconds. NMEA-0183 is a protocol that was designed for communication between marine electronics, including GPS devices.

NMEA messages are communicated to the Android framework from the GPS chip through the GPS Driver and GL Engine. This communication overhead introduces an offset between the time inside the NMEA message and the time the NMEA message is received in the Android Framework. This offset also varies and introduces jitter in the offset. The user application, that handles the received NMEA message in a callback, also suffers from latency. The user application handles the callback after a small delay, as Android is not a real-time OS.

The Android framework provides a time-stamp λ along with the received NMEA message. This time-stamp is the device's system time at the moment the Android framework received the NMEA message from the GL Engine.

The experimental application calculates the offset between the provided time-stamp in the callback and the GPS-time. This offset is the parsed time from the NMEA message minus the time-stamp that is provide in the callback.

The offset θ between the system clock and GPS-time ρ is denoted as:

$$\theta = \rho - \lambda \tag{4.1}$$

4.3.2 Network Time Protocol

NTP is a networking protocol for clock synchronisation between computer systems over packet-switched, variable-latency data networks.

To synchronize a client's clock with a remote server, the client must compute the round-trip delay time and the offset [34]. The round-trip delay δ is denoted as :

$$\delta = (t_4 - t_1) - (t_3 - t_2) \tag{4.2}$$

Where t_1 denotes the client's timestamp of the request-packet transmission, t_2 denotes the server's timestamp of the request-packet reception, t_3 denotes the server's timestamp of the response-packet transmission and t_4 denotes the client's timestamp of the response-packet reception. See Figure 4-2.

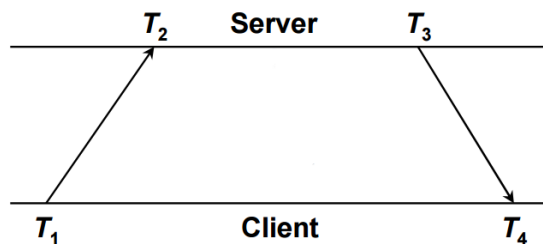


Figure 4-2: NTP time scheme. Source: [34]

Thus, $t_4 - t_1$ is the elapsed time on the client side between the emission of the request packet and the reception of the response packet and $t_3 - t_2$ is the time the server waited before sending the answer.

The offset between the system clock and NTP server clock θ is denoted as:

$$\theta = \frac{(t_2 - t_1) + (t_3 - t_4)}{2} \quad (4.3)$$

4.3.3 Measuring Clock Offsets

For both methods the offset θ is updated every second and stored as a global variable. For each detected sound-event e_j , the respective arrival time t_{ij} is corrected with offset θ . The recorded time for sound-event j at device i is thus denoted as:

$$t_{ij} = e_j - \theta \quad (4.4)$$

Measurement Set-Up

The results presented in this Section have been obtained by recording 500 NTP- and GPS offsets for fifteen Nexus 7 devices with an update rate of 1 *Hz*. The devices were placed outdoors with good line of sight to satellites (no obstructions in the form of buildings or trees). A local stand alone Wi-Fi network was utilised and the results were stored in a SQL database on a laptop. This laptop (Windows 8.1, Core-i7) also acted as the NTP time server.

Results

The averaged standard deviation σ for 15 devices is shown in Table 4.1

method	σ
GPS	8.32
NTP	27.16

Table 4.1: Mean standard deviation σ of the respective recorded offsets. The value for the NTP offsets has been obtained from a set of NTP offsets that was filtered from severe outliers.

Figure 4-3 shows a subset of 40 recorded NTP clock offsets from a single device. The Figure shows that the calculated offsets vary quite a bit and contain outliers.

The GPS clock offsets do not contain sever outliers but seem to have a sawtooth like waveform behaviour over longer periods, see Figure 4-4. It was made sure that the Android devices did not update their system clock during the offset recordings. It is unclear why the GPS offset value jumps with an interval of approximately five minutes. The experiment discussed in Section 4.8.1 raises the suspicion that the jumps do not occur at the same moment in time, this can cause worst case synchronisation errors between the devices of approximately 40 *ms*. This results in a worst case error of 14 *m*. This jump is either caused by the system clock or the GPS clock. Since the 5 minute interval is not seen in the NTP method, the jump must be caused by the clock of the GPS chip. Further investigation is required to see if this behaviour is device dependent, and how it can be optimised.

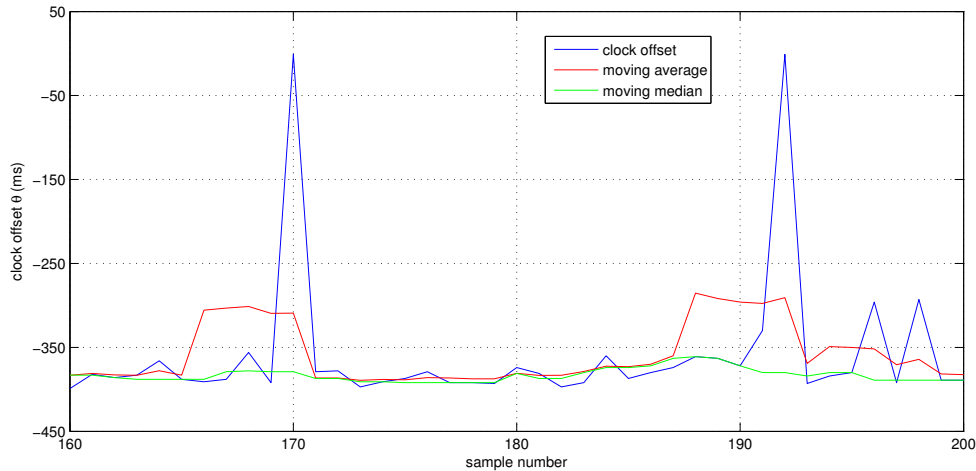


Figure 4-3: NTP clock offset, moving mean and median filtered signals. window-size = 5 samples

The behaviour described above was found after the outdoor localisation experiment was performed. Prior to the experiment only a small period of the GPS offsets was measured, which had a low standard deviation. Thus it was the preferred method and implemented during the experiment. The sawtooth behaviour is visible in later data validation experiments, as is discussed in Section 4.8.1. The CLASS Algorithm proposed in this thesis does however deal with this offset. This means that changing to a better time synchronisation method, the results can only improve.

It is important that the clock offset is stable and therefore the offsets need to be filtered. Figure 4-5 shows the standard deviation for the filtered clock offsets for both NTP and GPS time synchronisation. The sawtooth behaviour is not visible by looking at the standard deviation. Increasing the window size stabilizes the offset values, but a window that is too large will introduce error when the system clock drifts. A standard deviation of 2 *ms* is acceptable and thus implementing NTP synchronisation with a moving median filter and a window size of 200 would be a good choice in future experiments.

For GPS synchronisation it is important that there is a good GPS signal available. Since the experiments, discussed later in this thesis, took place outdoors this is not a problem. For indoor applications it is best to use NTP time synchronisation with a median moving average, or some other form of outlier detection.

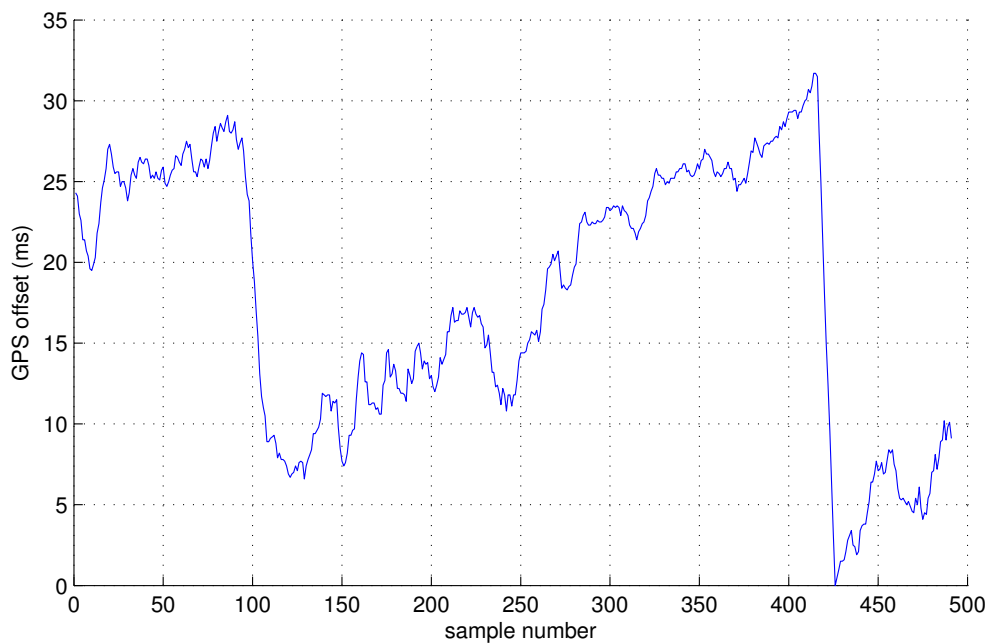


Figure 4-4: GPS offsets. Filtered with moving average. window-size = 5 samples. sample rate = $1Hz$

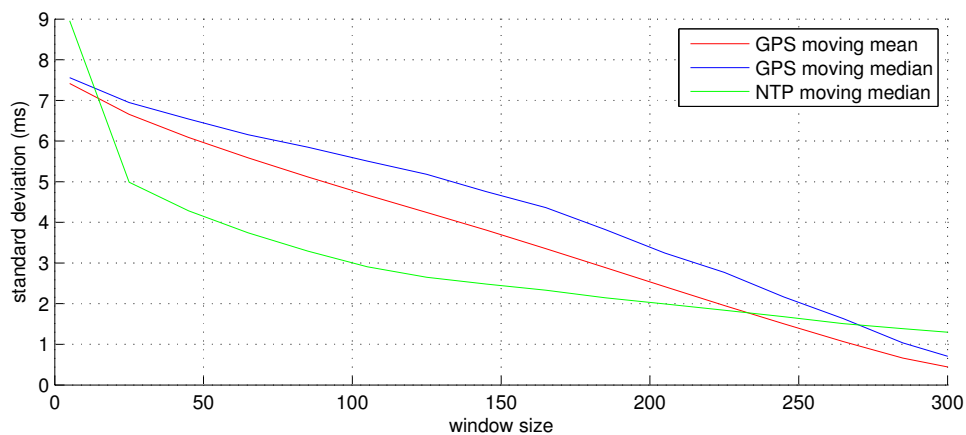


Figure 4-5: Standard deviation of clock offset, varying the window size for GPS and NTP

4.4 Android Audio Input Pipeline

Most Android devices contain at least one microphone which can be used to record audio. Each manufacturer implements its individual audio hardware components. In general it is not possible for an end user to optimize anything in the analog circuitry of Android devices. Android suffers from input latency which introduce errors in the sound detection time stamps. This Section describes the challenges encountered in Android that one encounters when developing high demand audio applications.

4.4.1 Input Latency

Recording a sound signal on a device includes the following steps: 1. the microphone picks up an audio signal from the environment 2. the recorded audio signal is processed by the Analog to Digital Converter (ADC) 3. the result of the ADC is stored in a buffer 4. when the buffer is full, the application processes the buffer whilst the ADC writes to a secondary buffer.

When playing a signal on the speaker of the device, similar steps occur in reverse order. The time it takes for an audio signal to travel from the microphone to the user application is called input latency. The time it takes for an audio signal to travel from the user application to the speaker is called output latency. The sum of these latencies is called round-trip latency.

Android needs to be universal and therefore high level APIs are used to support hundreds of thousands of different applications. Due to the overhead that occurs between the user application and the audio hardware, significant audio latencies are introduced. This latency is not constant and varies over time, this so called jitter causes unpredictable and unacceptable errors in ubiquitous applications that require accurate temporal information in relation to audio information.

Related Work

In the past many have investigated acoustic latency in operating systems, especially for personal computers [35–37]. Dannenberg et al. measured latency in various operating systems and found significant fractions of events that were delayed by tens of milliseconds. Under Windows 98 Dannenberg et al. found an increased audio latency for higher CPU loads [36]. For mobile devices, Mauerer et al. [25] describe an early overview on how to adapt Android 3.1 to a more real-time environment and demonstrate a latency measurement on a Motorola Xoom tablet.

Predicting Input Latency

In recent work, that was written during the preliminary research for this thesis, we tried to find a correlation between the system state of a device and it's input latency. When this correlation exists, it might be possible to predict and correct for the round-trip latency. In the resulting paper [38], we propose a framework that can quantify, predict and correct latencies in indeterministic devices. Through measurements on

multiple types of devices we found that Android devices suffer from input latency which can vary from approximately 50 *ms* to 150 *ms*, with a worst case jitter of 300 *ms* [38]. Except for a correlation between the device's price and the round-trip latency, it was not possible to find a high correlation between the state of the device and the audio latency, thus it is currently not possible to correct for audio latencies in smartphones. The proposed app that was developed during the research was not optimal at the time, this can be a reason for not finding the desired correlation. The application is currently being optimised so that a correlation might be found in future research. The tool that we developed can be used to gain an insight in audio latency of a device running Android OS and will be useful during the development of applications that are susceptible to audio latency.

Causes of Audio Latency

The first cause of latency is the analog circuitry of the device. According to Google engineers this does not always contribute to audio latency significantly [39].

Android is built on the Linux kernel which also contributes to the input latency. In [40] Rostedt describes how to find latencies in the Linux kernel with his latency tracer application "Ftrace". Rostedt works for Red Hat and maintains the stable Linux kernel releases of the real-time patch. He describes four different basic events that can cause latency within the Linux kernel:

1. Interrupts disabled - keeping interrupts from calling their handlers when a device triggers an interrupt to the CPU.
2. Pre-emption disabled - preventing a process that just woke up from running.
3. Scheduling latency - the time it takes a process to schedule in.
4. Interrupt inversion - the time that an interrupt handler is performing a task that is lower in priority than the task that it pre-empted.

According to Google the major surface-level contributors to audio latency in Android are the following [39]:

- Application
- Total number of buffers in pipeline
- Size of each buffer, in frames
- Additional latency after the app processor, such as from a Digital Signal Processor e.g. the FFT discussed in Section 4.4.3

Audio buffers are usually increased or decreased to overcome buffer under- and overruns. In Google's experience, the most common causes of under- and overruns include:

- the Linux CFS (Completely Fair Scheduler)
- high-priority threads with SCHED_FIFO scheduling
- priority inversion
- long scheduling latency
- long-running interrupt handlers
- long interrupt disable time
- power management
- security kernels

These contributors are arbitrary and are difficult to optimize on devices which run many different applications. Since a functional application should be able to work on any device, optimizing the experimental application for these contributors has not been taken into account.

It is hard to overcome the input-latency found in the Android OS and the resulting errors in the timestamps will have to be dealt with in the CLASS Algorithm, as discussed in Chapter 3.

4.4.2 OPEN-SL

In order to process the input data without overhead found in the Java environment, such as the memory garbage collector [41], the overhead-critical parts of the application have been developed in the Native Development Kit (NDK). The NDK is a tool set that allows developers to implement parts of an app using native-code languages such as C and C++ in the native layer, see Figure 4-1.

The Open Sound library (Open-SL) functions directly on top of the Android Hardware Abstraction Layer (HAL) and can be implemented to achieve lower latency audio [42].

4.4.3 Audio Input Filter

In order to distinguish sound-events from environmental noise, such as wind blowing in the microphone, it is mandatory to filter the audio input. Since sound events can be distinguished by their frequency in a certain range, it makes sense to filter the input with a band-pass filter. Noise in other frequency ranges is filtered out and the number of false triggers in sound-event detection is decreased.

Filtering can be done in the time domain, e.g. by use of a Finite Impulse Response (FIR) filter, or in the frequency domain, by implementing a FFT. A FFT filter requires more calculation steps than a FIR filter but is more flexible in the configuration. A FIR filter requires a recalculation of all the coefficients when changing the cut-off frequencies. Transforming the input signal to the frequency domain opens up the ability to detect audio-events by their frequency signature. For these reasons an FFT filter was implemented in the experimental application.

FFT Implementation

The Fast Fourier Transform efficiently computes the discrete Fourier Transform of a signal and results in a set of magnitudes and phases ordered by their frequency. When the input data is converted to the frequency domain we have $\frac{s_f}{2}$ magnitudes and phases available in bins. Where s_f is the FFT size.

The magnitude of each bin represents the power information for this frequency band. The frequency resolution of the FFT f_r (width of the bins) is calculated with the following Equation:

$$f_r = \frac{f_s}{2} \cdot \frac{2}{s_f} Hz \quad (4.5)$$

Where f_s denotes the sampling rate and s_f the FFT size.

The FFT introduces a delay in the input signal since it needs a number of audio samples to perform the Fourier Transformation.

In the frequency domain we can cut away frequencies that we don't want, thus create a band-pass filter. The filtered sound signal is then transformed back to the time domain.

4.4.4 Determining the Desired Frequency Range

The frequency band of the horn signal is wide. Figure 4-6 displays the frequency spectrum for environmental noise with no horn a recording where the air horn is sounded and the difference between these two spectra. Both sound samples were recorded with a sampling rate of $44.1 kHz$. The spectra without the signal is calculated from a $2 : 30 min$ recording that includes ambient sounds such as wind blowing in the microphone, cars and talking people. In order to minimize the error introduced by environmental noise, whilst maintaining a good signal to noise ratio, the band pass filter in the experimental application is configured between 17 and $18 kHz$.

4.5 Sound-Event Detection

In the experiment set-up the sound of an air horn is used as a distinguished sound-event. A sound-event is detected when the energy signal in the frequency band of the air horn exceeds the threshold value λ , see Section 4.5.1. A FFT solution has been implemented in the current application and this can be utilised to improve the matching Algorithm. Frequency patterns can be found in the environment and used as a sound-event. An optimised matching Algorithm can later be investigated in order to use arbitrary ambient sound as distinguished sound-events.

The input signal might contain noise in the form of high energy peaks. Therefore the energy value of at least N audio samples needs to be averaged in order to filter out noisy peaks. Averaging samples decreases the resolution of event detection, therefore the value of N directly impacts the error in the time-stamp.

E.g. with a sample rate f_s of $44.1 kHz$ and N set to 512 samples, this relates to a resolution of approximately $12 ms$. A device with a small microphone gain might

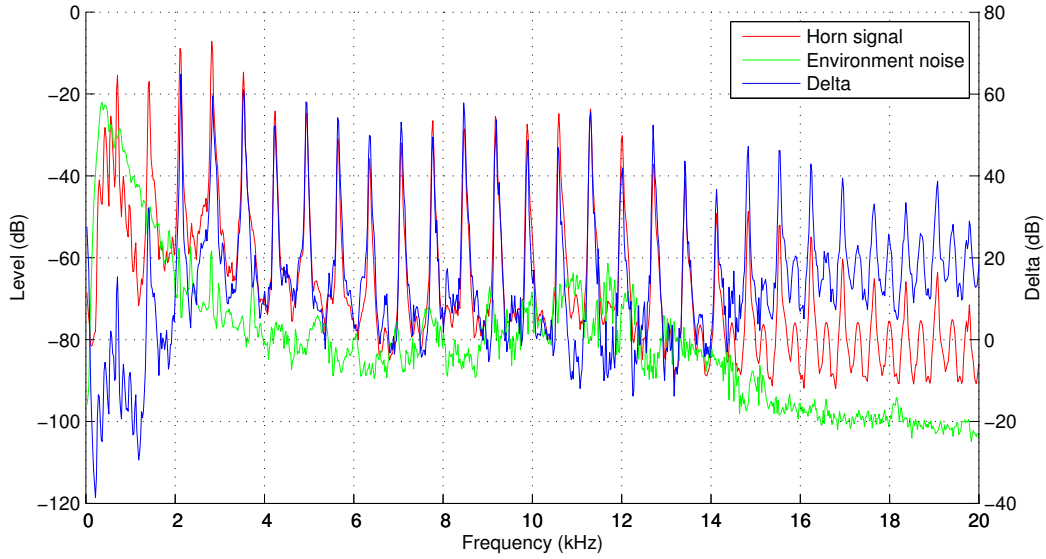


Figure 4-6: Frequency spectra

measure an average value that is still below the threshold whilst a device with a higher gain will measure this frame to be higher than the threshold. This results in an error of at least 12 *ms* in the measured time between the phones. Therefore the threshold needs to be determined per device and the size of N should not be too large.

4.5.1 Deriving the Desired Threshold Value

Although all devices in the experimental set-up are of the same build, the gain in the amplification of the input signal differs for each device. It is problematic to use a static threshold because various models will measure different energy levels for the same sound at an identical distance.

The threshold λ is determined by 'listening' to ambient sound for G samples. This allows for a threshold value that is proportional to the device's microphone gain. See Equation 4.6

$$\lambda = \frac{SNR}{G} \sum_{g=1}^G x_g \quad (4.6)$$

Where x denotes the input signal and the SNR is a value between 0 and 1. In the experimental set-up, G is set to a value so that the devices listen to the environment for three seconds.

4.5.2 Averaging Input Samples

Averaging a smaller amount of samples (N) results in a higher resolution in the threshold detection but increases CPU load.

When we want to average the input signal for three periods $3 \cdot T_r$ of the expected reference signal f_r (the horn), we can derive N . See Equation 4.7.

$$N = \left\lceil \frac{3T_r}{T_s} \right\rceil \quad (4.7)$$

Where T_s denotes the sampling period $\frac{1}{f_s}$. E.g. when a distinguishable sound-event has a base frequency of 17.5 kHz and the device uses a sampling rate of 44.1 kHz , N is 8. The resolution for input sampling then becomes $181 \mu\text{s}$, which is well within the standard deviation of other errors introduced by the overall application.

4.6 Time-Stamping

In order to localise all the nodes and sound locations, each device records a time stamp when it overhears a distinguished event in the environment. These time-stamps are aggregated in a central location where the CLASS Algorithm is executed, see Chapter 3. In order to acquire a time stamp the system clock, located in the Linux kernel, must be accessed from the Android Application layer. This can add an additional offset in the time-stamp due to OS overhead. This has been accounted for by developing time-critical parts of the application in the Android Native Layer, which is less susceptible to overall OS overhead.

The FFT filter described in Section 4.4.3 introduces an additional delay, and possibly jitter. Since we can only determine the sound-event's presence in the input signal after the FFT processing, there will be an additional error in the time-stamp. The time stamps itself are therefore buffered. When an audio input buffer is received, a time stamp is recorded and pushed into a First In First Out (FIFO) buffer. The FFT buffer size is equal to the audio buffer size. When a sound-event is detected, the time-stamp that corresponds to the FFT-output buffer is recovered from the FIFO buffer and used as time of arrival d_{ij} .

4.7 Networking

In order to aggregate the time-stamps, and disperse other information between devices, a network of some sort is required. In the experimental set-up a WiFi-router is utilised to create a local infrastructure. In future works an ad-hoc network can be implemented so that the approach does not need any other infrastructure than the network of devices itself. In [43] Turkes et. al. present Cocoon, a data dissemination model for smartphones that requires no existing infrastructure. Cocoon is an interesting application to combine with the work in this thesis.

4.7.1 Server

In future works a smart-phone can be implemented as a server node that collects the time-stamps and executes the CLASS Algorithm. In the experimental application a

Laptop acts as the server node in the local network. An Apache server is deployed on a laptop which is connected to a MySQL database. All devices connect to the server through the Wi-Fi network and transmit data by the HTTP POST request method. The server stores the data in the appropriate Table. The following data is stored in the database:

1. sound-event identifier
2. device identifier (Media Access Control (MAC) address)
3. time-stamp of the sound-event
4. sound-event location

Items 1, 2, 3 are mandatory for the CLASS Algorithm. The sound-event location (4) is recorded in order to link the measurements to the ground truth. Matlab connects to the database and accesses the data through SQL scripting to generate TOA matrix T .

4.8 Data Quality

The experimental application has been used to collect real measurement data in an outdoor environment. This Section describes two different experiments that were conducted to assess the quality of data acquisition by the Android Application.

Section 4.8.1 describes a measurement where all devices were located at an identical location in order to assess overall error in the measurements. Section 4.8.2 describes an experiment that estimated several distances by using a sound signal in a straight line over multiple devices.

4.8.1 Placing all Devices at an Identical Location

In order to determine the total error that is introduced by audio latency and synchronisation offsets, measurements were recorded with all devices located at an identical location.

Measurement Set-Up

The devices were placed within in an area of 1 m^2 . 50 sound-events were generated directly above this area with an air horn and the TOA values were recorded.

The TDOA between device i and k for sound-event j is calculated as follows:

$$t_{ijk} = t_{ij} - t_{kj} \quad (4.8)$$

Where t_{kj} resembles the reference device. See Section 3.1.1 for a more detailed explanation regarding the TDOA calculation. The TDOA values in this experiment were calculated between all pair of devices.

Results when Synchronising with GPS

The moving average window size was set to 10 samples when recording the GPS offsets presented in this Section. From the results derived in Section 4.3.3 we would expect a time-synchronisation error with a worst case deviation of 7.5 ms . Over 50 measurements the standard deviation, when synchronising with GPS, varies between 6 ms and 13 ms , with a peak at 10 ms , see Figure 4-7. This standard deviation includes the error that is introduced by audio latency and other overhead in the Android OS.

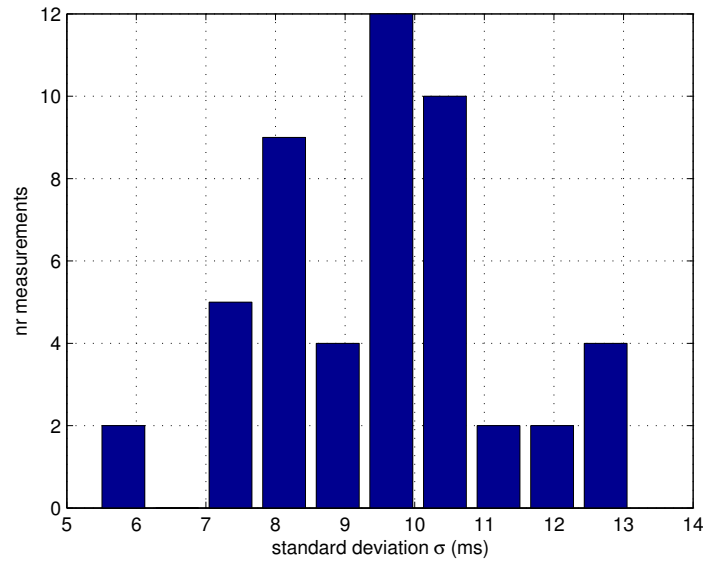


Figure 4-7: Standard deviation σ of the TDOA values when synchronised by GPS

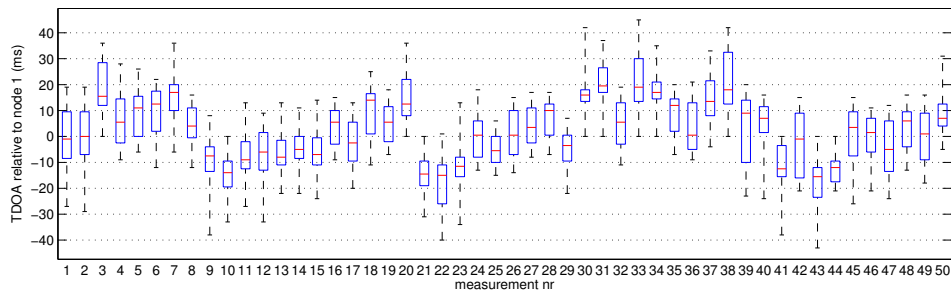


Figure 4-8: Boxplot of TDOA values relative to device 1

The measurements shown in Figure 4-8 have been taken with an approximate interval of roughly 30 sec . There seems to be a pattern in the offset relative to device 1. Figure 4-9 shows the mean TDOA value relative to three different devices k . Although the data in Figure 4-9 also contains the audio latency and other overhead errors, the pattern seems correlated with the GPS synchronisation offset error as depicted in

Figure 4-4. The mean TDOA values are shifted in time when different devices are set as reference device k , see Figure 4-9. Between all 16 devices that were used during this measurement a worst-case time synchronisation offset of approximately 20 ms can occur for some TDOA pairs. Errors in TDOA values are however filtered by the CLASS Algorithm described in Section 3.5.

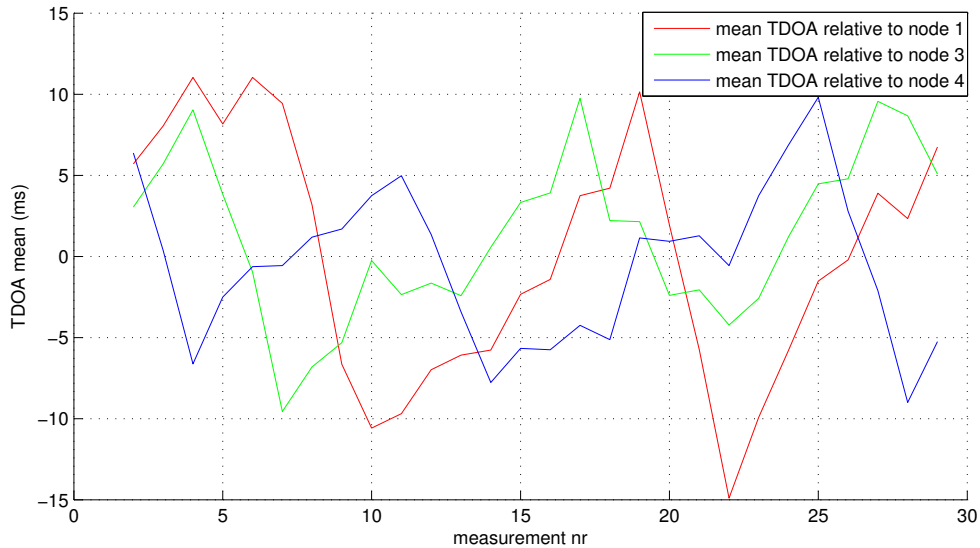


Figure 4-9: Mean TDOA values, relative to different nodes. An optimal TDOA should be near 0, since all devices are practically at an identical location.

Results when Synchronising with NTP

In order to investigate if improvements could be made by using NTP synchronisation, measurements were made with optimal values for NTP synchronisation. Based on the results found in Section 4.3.3, a moving median filter with a window size of 200 samples was implemented to obtain the results presented in this Section. We would expect a time-synchronisation error with a worst case deviation of 2 ms . Over 50 measurements the standard deviation, when synchronising with NTP, varies between 6.7 ms and 11.3 ms , with a peak at 7.3 ms , see Figure 4-10. This standard deviation includes the error that is introduced by audio latency and other overhead in the Android OS. Figure 4-11 displays the box plot results for all TDOA values per measurement. The median values are distributed around the zero value.

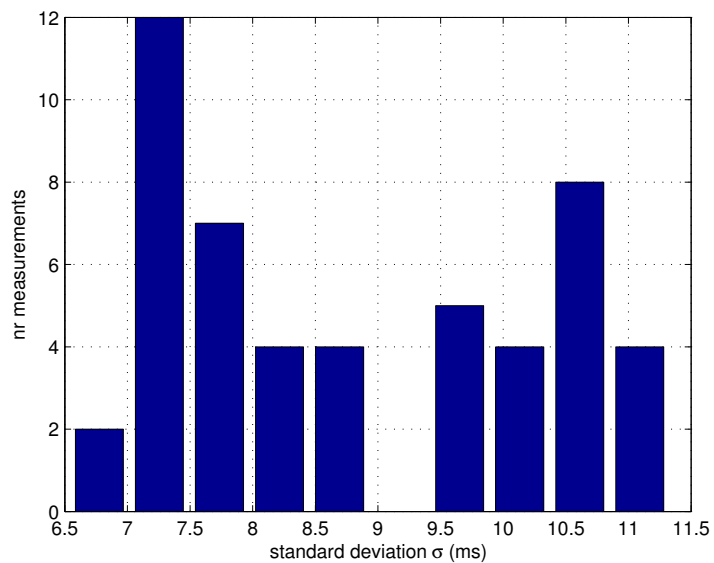


Figure 4-10: Standard deviation σ of the TDOA values when synchronized by NTP.

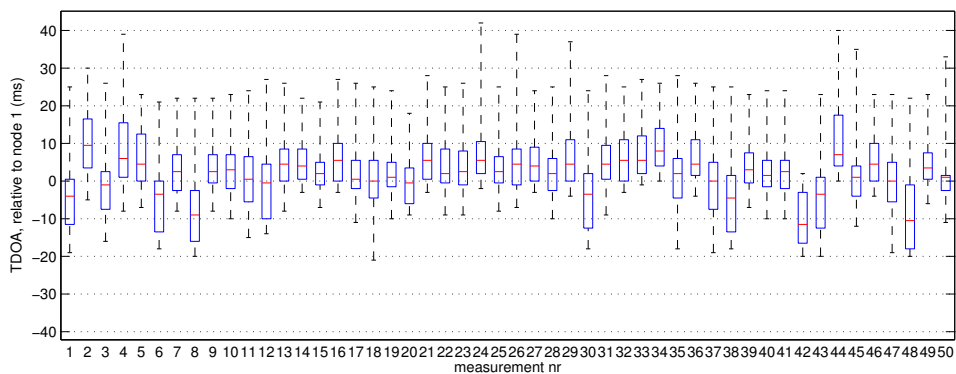


Figure 4-11: Boxplot of TDOA values relative to device 1

Comparison

In order to investigate the possibility of improvements, this Section compares NTP synchronisation to the implemented GPS method. The GPS method used a moving average with a windows size of 10 samples, whilst the NTP method used a moving median with a window size of 200 samples.

When comparing the standard deviation between the two synchronisation methods, some outliers are observed for the GPS method, see Figure 4-12. These outliers are expected to originate from the worst case offset between the devices due to the 5 minute interval reset that occurs when synchronising with GPS, see Section 4.3.3. Table 4.2 depicts the the mean standard deviation of TDOA values for NTP and GPS respectively. The standard deviation for the GPS method is a bit higher than the NTP method, this is due to the outliers found in the GPS method. The standard deviation in the results for synchronisation through NTP is lower and the median values shown in Figure 4-11 are more stable than the results for synchronisation with GPS (Figure 4-8). NTP seems to have better accuracy than GPS, this is however caused by the 5 minute update in GPS synchronisation that requires more investigation. The results suggest that some improvement could be made by implementing NTP synchronisation.

Method	$\sigma(ms)$
GPS	9.48
NTP	8.79

Table 4.2: σ for GPS and NTP synchronisation

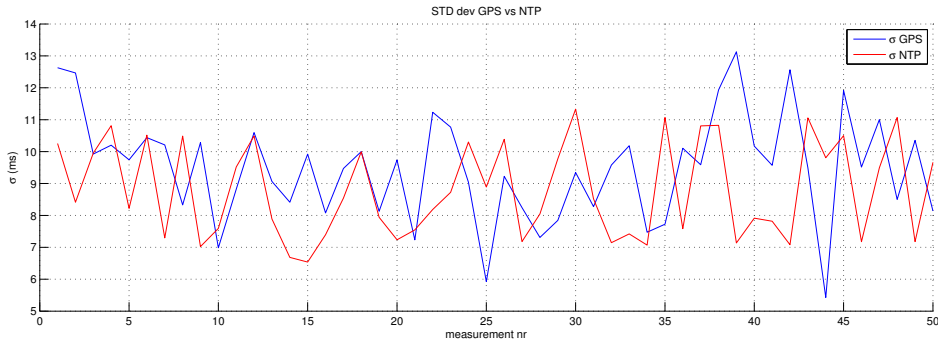


Figure 4-12: Mean standard deviation σ of TDOA values per measurement for GPS and NTP synchronisation method respectively

4.8.2 Distance Measurement

In order to validate the estimation of distances based on TDOA measurements a distance experiment was preformed. Several distances were estimated by using a sound

signal in a straight line over multiple devices and recording the TDOA differences between groups of devices at different locations. The experiment is performed for both GPS and NTP synchronisation methods to investigate the differences.

Measurement Set-Up

The measurement was performed outdoors in a windy environment. The sound-events were generated with an air horn. The devices were placed on tripods at a height of approximately 1 m. All devices were grouped in rows and placed as close to each other as possible in each row. The shape of the constellation is shown in Figure 4-13. 50 measurements were taken on 4 devices for each distance, thus a total of 200 samples were recorded per distance

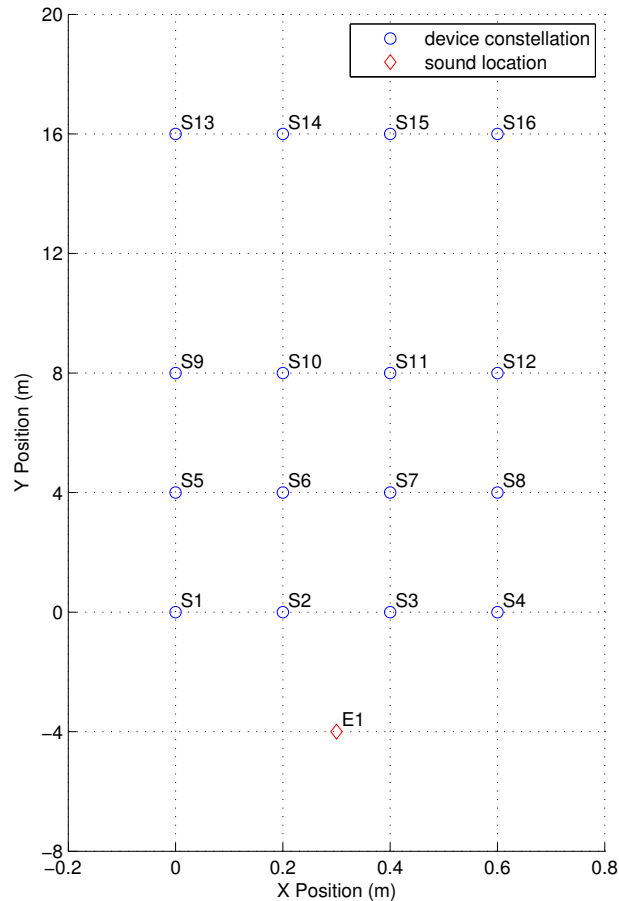


Figure 4-13: 2D overview of the measurement set-up

Results when Synchronising with GPS

Box plots of the measured distances are presented in Figure 4-14a. The median values are close to the ground truth distance. Some outliers are found for the measured distances that are further away from the sound source. Distances are perceived as outliers if they are larger than $q_3 + w \cdot (q_3 - q_1)$ or smaller than $q_1 - w \cdot (q_3 - q_1)$, where q_1 and q_3 are the 25 th and 75 th percentiles, respectively and $w = 2$.

During this measurement the first 8 m of the set-up was protected from wind by bushes, whilst the remaining 8 m was in pretty strong winds. This might have caused a few more outliers in the rear of the measurement set-up due to errors in the calibration phase. The threshold for the devices in the back might have been higher than devices in the front, which requires a higher volume that is obtained later in time due to onset time in the sound signal.

Table 4.3 shows the respective results that were obtained from the measurement when synchronised by GPS. The mean values are more accurate than the median values. The results show that the measured distances have a normal distribution around the ground truth.

ground truth distance	4 m	4 m	8 m	12 m	16 m
min	-7.49	-6.47	-6.47	-5.44	6.13
mean	4.15	4.53	7.21	11.74	15.89
max	16.33	15.65	46.28	59.55	58.87
0.25 quantile	0.68	1.02	3.06	8.34	11.06
median	3.57	4.08	6.64	10.89	14.97
0.75 quantile	7.83	7.15	10.89	14.63	18.38

Table 4.3: Box plot variables of measured distances for synchronisation by GPS

Results when Synchronising with NTP

Box plots of the measured distances are presented in Figure 4-14b. Distances are perceived as outliers if they are larger than $q_3 + w \cdot (q_3 - q_1)$ or smaller than $q_1 - w \cdot (q_3 - q_1)$, where q_1 and q_3 are the 25 th and 75 th percentiles, respectively and $w = 2$.

Table 4.4 shows the respective results that were obtained from the measurement when synchronised by NTP. The mean values are more accurate than the median values. The results show that the measured distances have a normal distribution around the ground truth. For almost all distances the TDOA values are larger than the ground truth.

Comparison

Figure 4-14 shows the measured distance results for both synchronisation methods. The estimations with NTP synchronisation have a lower standard deviation than GPS, this is as expected based on the results found in Section 4.8.1.

ground truth distance	4 m	4 m	8 m	12 m	16 m
min	-2.72	-2.72	0.34	5.10	8.51
mean	4.51	3.86	8.48	12.34	16.84
max	10.89	11.57	15.65	17.70	24.16
0.25 quantile	2.04	2.04	4.76	9.53	13.95
median	5.10	3.06	8.68	13.27	17.01
0.75 quantile	6.47	4.08	11.57	14.97	19.74

Table 4.4: Box plot variables of measured distances for synchronisation by NTP

Table 4.5 shows the error in the estimated distance for both methods respectively. The error is the difference between the mean estimated value and the ground truth. The error in metres, for the mean measured distances, do not differ much. The mean difference is 9 *cm*. During the acquirement of the data-set, that was used to obtain the final solution set in this thesis, GPS synchronisation was used. According to the results presented in this Section, some improvement can be expected in the localisation results by changing the time synchronisation method. Less TDOA values at identical locations are likely to be filtered out, which could improve the overall accuracy. It must be noted that solving the problem with the 5 minute update period, discussed in Section 4.8.1, could improve the GPS synchronisation method, which implies that a new comparison would have to be made.

	4 m	4 m	8 m	12 m	16 m	mean
error GPS (m)	0.15	0.53	0.79	0.26	0.11	0.37
error NTP (m)	0.51	0.14	0.48	0.34	0.84	0.46

Table 4.5: Mean error for GPS and NTP synchronisation

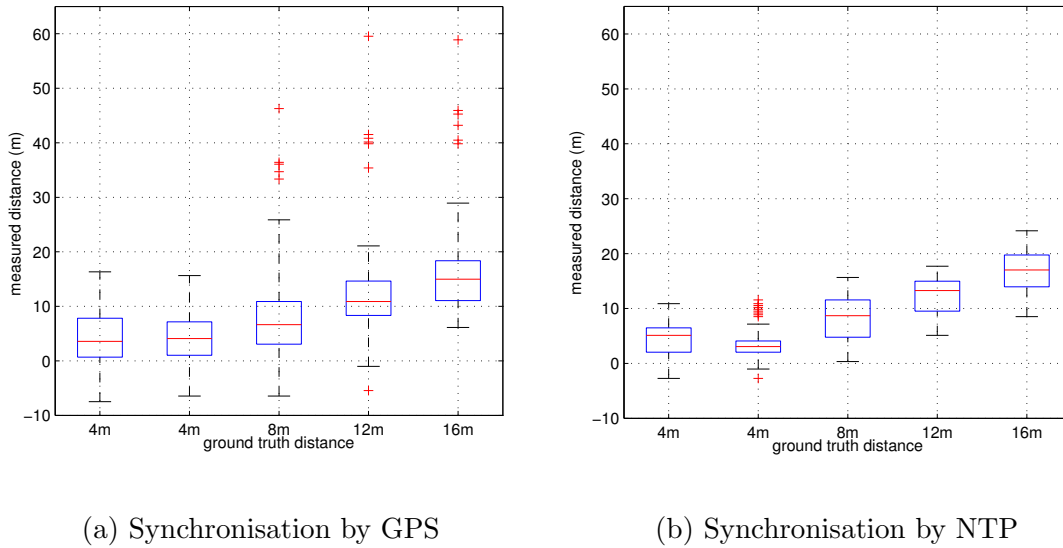


Figure 4-14: Comparison of measured distances between GPS and NTP synchronisation methods, respectively

4.9 Conclusion

This Chapter described a technical analysis of the Android OS in relation with sound localisation accuracy and discussed the acquisition of sound-event data in a network of mobile devices. The following technical limitations of Android, ordered from largest to smallest contributor to inaccuracies, have been identified and discussed:

1. Audio input latency

The biggest contributors for inaccuracies in TDOA measurements are audio input latency and poor time synchronisation. The audio input latency differs per device and can have worst case jitters of 300 ms . It is hard to overcome the input-latency found in the Android OS and the resulting error in TDOA values will have to be dealt with in the CLASS Algorithm, as discussed in Chapter 3.

2. Poor time synchronisation

Time synchronisation would be considered to be the second largest contributor to TDOA measurement inaccuracies. The standard deviation in system clock offsets can be reduced to 2 ms by implementing a moving mean- or median filter with a windows size of 200 samples. An internal clock update with a 5 min interval can introduce a worst case offset of 40 ms in GPS synchronisation. In order to find the exact cause of this clock-update, further investigation is required. The difference in distance measurement results between GPS and NTP time synchronisation is not very big. Improving the time synchronisation method will reduce the number of outliers in TDOA measurements.

3. Difference in microphone gain per device

The difference in microphone gain can cause a device to detect a sound-event earlier or later than other devices, which will cause an offset in the respective TDOA measurement for those devices. In the experimental application this is accounted for by using a variable threshold that is depended on the microphone gain of each respective device and utilising sound-events with short onset times. Other techniques, such as cross correlation of sound-events in the frequency spectrum will probably not suffer from this problem and can be investigated in future work.

4. Delays in recording time-stamps

In order to acquire a time stamp the system clock, located in the Linux kernel, must be accessed from the Android Application layer. This can add an additional offset in the time-stamp due to OS overhead. This has been accounted for by developing time-critical parts of the application in the Android Native Layer, which is less susceptible to overall OS overhead.

5. Implementing a DSP like FFT

A FFT filter adds a little offset in the time-stamp, but this can be accounted for by buffering the time-stamps and should not contribute significantly to the error in TDOA measurements.

6. Noise in the form of peaks in the microphone signal

Noisy peaks that occur at the input of the device can trigger false sound-event detections. This is accounted for by averaging input values whilst making sure the detection resolution remains accurate enough.

All devices in a Cooperative Network detect an identical sound-event by measuring the energy level in a preset frequency band. When the energy exceeds some threshold λ , a sound-event is detected. In the experiment set-up the sound of an air horn is used as a distinguished sound-event. A sound-event is detected when the energy signal in the frequency band of the air horn exceeds the threshold value λ , see Section 4.5.1. A FFT solution has been implemented in the current application and this can be utilised to improve the matching Algorithm. Frequency patterns can be found in the environment and used as a sound-event. An optimised matching Algorithm can later be investigated in order to use arbitrary ambient sound as distinguished sound-events.

The minimum information that needs to be shared among devices in a Cooperative Network for the CLASS Algorithm to work are: (i) a sound-event identifier, (ii) a device identifier, (iii) the time-stamp of the sound-event arrival time.

When measuring TDOA values for multiple distances in a straight line, a mean error of 37 *cm* can be obtained. This error includes all contributors to inaccuracies that have been listed above.

Now that the accuracy of the Application and related TDOA data set is known Chapter 5 will assess the performance and behaviour of the CLASS Algorithm.

Chapter 5

Experimental Validation

This Chapter presents the localisation results that have been obtained by executing the CLASS Algorithm described in Chapter 3. The CLASS Algorithm was executed ten times. Twenty sound-events were generated at locations around the device constellation. At each location four sound-events were generated. Sixteen devices were placed in a $12\text{ m} \times 12\text{ m}$ grid with a perpendicular inter-device distance of 4 m .

A mean RMSE of 2.4 m with a standard deviation of 0.21 m is achieved. The mean RMSE of the estimated directions is 76.24° with a standard deviation of 1.28° . A solution set for the relative positions of sixteen devices is plotted in Figure 5-1. The RMSE for this particular solution set is 2.08 m . Figure 5-1 displays a top-down 2D overview of the ground truth for device locations S . The estimated locations for each device are represented as blue dots. The localisation errors are indicated by red lines which represent the euclidean distance.

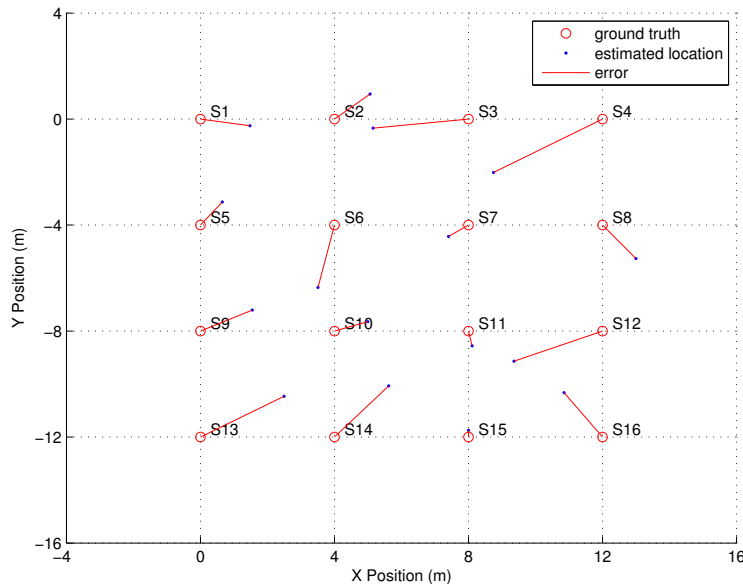


Figure 5-1: Localisation results. The RMSE is 2.08 m

The solution that is returned by the CLASS Algorithm is rotated and translated to align with the ground truth, see Section 2.6. Some devices are located further away from their ground truth than others. The errors in TDOA, for all sound-events, are represented in the relative positions shown in Figure 5-1. The devices with the larger error will have the worst errors in their respective TDOA values. An error introduced by one device will always influence the others, as TDOA values are always relative.

Most estimated locations tend to move to the centre of the constellation. This is a result of the HBOS Algorithm that is applied to TDOA measurements at an identical location, see Section 3.5. As discussed in Section 3.5 this Algorithm is not optimal when only four TDOA measurements are available. When no TDOA inliers are found between a pair of nodes, the respective value will be set to zero, see Algorithm 1. This can result in a TDOA matrix that contains a lot of zero values, which represent a solution where devices are placed on top of each other. The centroid of the solution set equals that of the ground truth, thus when all devices are placed on top of each other, the estimated position for the devices will be in the centre. This result indicates that the TDOA filter requires more elaboration in future works. Instead of setting the respective value to zero when the measurement is ambiguous, a value should be calculated and given a low weight. The least squares problem described in Section 3.3 should be updated to a weighted least squares problem so that each filtered TDOA value Δ_{ijk} is weighted. TDOA measurements with few inliers will have little influence on the results, whilst measurements with a lot of inliers will have a strong influence on the results.

A solution set of the estimated directions towards the sound-event origins is shown in Figure 5-2. Figure 5-2 displays a top-down 2D overview of the ground truth for sound-event origins E . The device constellation is not shown in this Figure for sake of clarity. Each direction α_j is plotted as a vector that originates at the centre of the device constellation and is oriented towards sound-event E_j . The angular RMSE for this particular solution set is 75.56° . The solution that is returned by the CLASS Algorithm is rotated and translated to align with the ground truth, see Section 2.6.

The results indicate that the directions are accurate enough to use as indicators to sound-event origins. An absolute position or direction must be known in order to find the true direction towards the origin of a sound-event or device. This can be done by knowing the location of at least two devices and/or sound-event locations. The exact location of the sound-event can be triangulated with the TDOA data after the device locations have been estimated. This requires an additional step in the CLASS Algorithm.

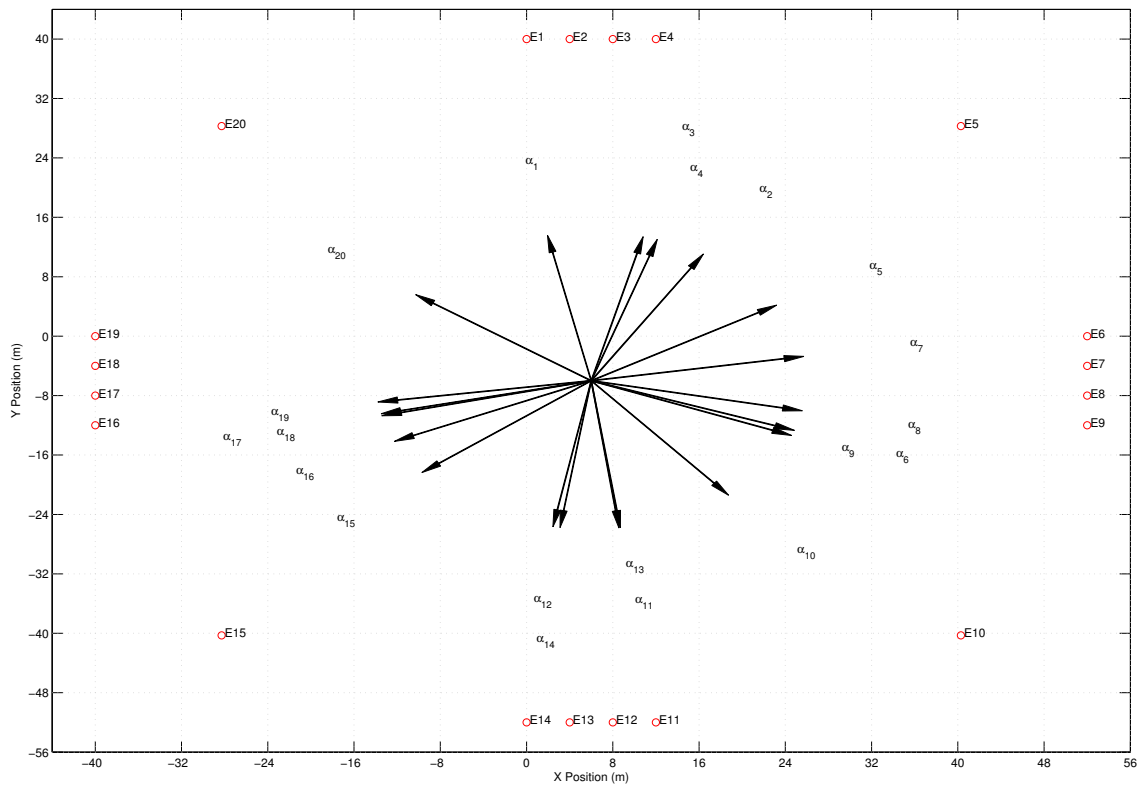


Figure 5-2: Estimation of directions towards sound-event origins. The angular RMSE is 75.56°

Chapter 6

Conclusion

In the introduction of this thesis we asked the following question:

When utilising only environmental sound that originates from unknown positions, what techniques for Cooperative Localisation can be used on Android devices that can achieve accuracy within several metres, and what factors will influence this accuracy?

In order to answer this question the CLASS Algorithm was introduced. This Algorithm produces a location set for all devices and a set of directions towards the origins of the sound-events. The CLASS Algorithm is able to localise a set of devices with a mean RMSE of 2.4 *m* with a standard deviation of 0.21 *m*. The mean RMSE of the estimated directions is 76.24° with a standard deviation of 1.28°. The main factors that influence this accuracy are inherent in the technical limitations of utilising a non RT OS, see sub-question 2 below.

The following sub-questions were derived from the main question:

1. *What technique localises devices in a network without prior knowledge of the device locations and sound signal origins?*

The relation between the position of devices X , the directions towards the origins of the sound-event locations α and the TDOA values was described. TDOA values that are inliers are averaged for sound-events originating from an identical location in a matrix Δ . The Levenberg-Marquardt solver is utilised to find an optimal solution set for the all the positions and directions from Δ . Finally a HBOS Algorithm is implemented to filter outliers from the localisation results.

2. *What are the technical limitations of utilising a non Real-Time OS like Android for Cooperative Localisation that achieves accuracy within several metres?*

The main limitation of the Android OS for it's use in Cooperative Localisation by sound is its input latency. Input latency in Android suffers from large variations that cannot be predicted and corrected for. The audio input latency differs per device and can have worst case jitters of 300 *ms*. It is hard to overcome

the input latency and the resulting error in TDOA measurements will have to be dealt with in the CLASS Algorithm, as discussed in Chapter 3.

The following technical limitations of Android, ordered from largest to smallest contributors to TDOA inaccuracies, have been identified and discussed: (i) audio input latency, (ii) poor time synchronisation, (iii) difference in microphone gain per device, (iv) delays in recording time-stamps, (v) implementing a DSP like FFT, (vi) noise in the form of peaks in the microphone signal.

3. *What are the consequences of inaccurate TDOA measurements for Cooperative Localisation and how can they be minimised to achieve accuracy within several metres?*

Inaccurate TDOA measurements can cause the solver to find localisation solutions that have very large errors. The CLASS Algorithm deals with inaccurate measurement data by finding and averaging TDOA values, and localisation results, that are inliers. Improving time synchronisation accuracy will reduce the number of outliers in TDOA measurements. When measuring TDOA values for multiple distances in a straight line, a mean error of 37 *cm* can be obtained.

4. *What techniques can be used to simultaneously detect an identical sound signal on all devices in a Cooperative Network?*

All devices in a Cooperative Network detect an identical sound-event by measuring the energy level in a preset frequency band. When the energy exceeds some threshold λ , a sound-event is detected. Further research is required to elaborate on this question, see Section 6.1.

5. *What type of sound signals can be used for Cooperative Localisation?*

The experimental application utilised the sound of an air horn as a distinguished sound-event. The implemented threshold matching algorithm requires sound signals that have a short onset time and a good SNR. A FFT solution has been implemented in the current application and this can be utilised to improve the matching Algorithm. Frequency patterns can be found in the environment and used as a sound-event. An optimised matching Algorithm can later be investigated in order to use arbitrary ambient sound as distinguished sound-events.

6. *What techniques can be used to share information in a Cooperative Network without using existing infrastructure?*

The experimental application described in this thesis utilised a local Wi-Fi network to route TDOA measurement data to a server. In future works an ad-hoc network can be implemented so that the approach does not need any other infrastructure than the network of devices itself. In [43] Turkes et. al. present Cocoon, a data dissemination model for smartphones that requires no existing infrastructure. Cocoon is an interesting application to combine with the work in this thesis.

7. *What kind of information do devices need to share within the network for Co-operative Localisation by sound?*

The minimum data that need to be shared for the CLASS Algorithm to work are: (i) a sound-event identifier, (ii) device identifier, (iii) time-stamp of the sound-event.

The steps described in Section 6.1 gear towards a pervasive application that allows a network of devices to localise themselves, and the origins of sound in the environment, without the use of any existing infrastructure. The fact that this application runs on Android devices, available and used extensively around the world, will allow the application to be used in various contexts.

6.1 Future Work

Improvements can be made in the Android Application to improve the accuracy of data acquisition. The five minute interval of the system clock update found in GPS synchronisation requires more investigation.

An alternative outlier detection method for TDOA values at identical locations should be investigated. Instead of setting the respective value to zero when the measurement is ambiguous, a value should be calculated and given a low weight. The least squares problem described in Section 3.3 should be updated to a weighted least squares problem.

Outlier removal can be improved in future work by expanding the HBOS Algorithm with an Algorithm similar to the one described in [9]. This requires more steps in the CLASS Algorithm but will improve the localisation accuracy. Currently only the angles toward the sound-event origins are estimated, this can be elaborated to pinpoint the exact location of the sound-event. When the CLASS Algorithm is elaborated it should be compared to other methods.

It would be interesting to confirm the minimum required distance between the sound-events and the device constellation, for the FFA to hold. This can be done by conducting an experiment where the radius of the circle, along which the sound-event locations are distributed, is decreased incrementally.

The Android application and server configuration are close to a working demo set-up. The work can be expanded so that a live demo application can be developed. The demo can utilise a personal computer to collect the TOA data and preform the CLASS Algorithm. The next step would be to implement the server side on a smartphone. It is interesting to investigate the integration of Google's Ceres solver [44] in the Android application.

The matching Algorithm described in Section 4.5 can be elaborated in future works. A FFT solution has been implemented in the current application and this can be utilised to improve the matching Algorithm. Frequency patterns can be found in the environment and used as a sound-event. An interesting question for future work then becomes: *What techniques can extract multiple sound-events from longer repet-*

itive sound so that they can be used for Cooperative Localisation, and how accurately can these events be distinguished from each other?

An ad-hoc network can be implemented so that the approach does not need any other infrastructure than the network of devices itself. In [43] Turkes et. al. present Cocoon, a data dissemination application for smartphones that requires no existing infrastructure. Cocoon is an interesting application to combine with Cooperative Localisation. An example for a research question would be: *How to utilise an ad-hoc network in Cooperative Localisation by Sound?*

Bibliography

- [1] Reza Zekavat and R Michael Buehrer. *Handbook of position location: Theory, practice and advances*, volume 27. John Wiley & Sons, 2011.
- [2] Amitangshu Pal. Localization Algorithms in Wireless Sensor Networks: Current Approaches and Future Challenges. *Network Protocols and Algorithms*, 2(1):45–73, 2010.
- [3] DoD. Global Positioning System Standard Positioning Service. (September):1 – 160, 2008.
- [4] Henk Wymeersch, Jaime Lien, and Moe Z. Win. Cooperative localization in wireless networks. *Proceedings of the IEEE*, 97(2):427–450, 2009.
- [5] R. Biswas and S. Thrun. A passive approach to sensor network localization. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2, 2004.
- [6] Sebastian Thrun. Affine Structure From Sound. *Conference on Neural Information Processing Systems*, 18:1353–1360, 2005.
- [7] Johannes Wendeberg, Thomas Janson, and Christian Schindelbauer. Self-Localization based on ambient signals. *Theoretical Computer Science*, 453:98–109, 2012.
- [8] Johannes Wendeberg, Fabian Höflinger, Christian Schindelbauer, and Leonhard Reindl. Calibration-free TDOA self-localisation. *Journal of Location Based Services*, 7(January 2015):121–144, 2013.
- [9] Simon Burgess, Yubin Kuang, and Kalle Å ström. TOA sensor network self-calibration for receiver and transmitter spaces with difference in dimension. *Signal Processing*, 107:33–42, 2014.
- [10] Guoqiang Mao, Bar Fidan, and Brian D.O. Anderson. Wireless sensor network localization techniques. *Computer Networks*, 51:2529–2553, 2007.
- [11] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The Anatomy of a Context-Aware Application. 1, 2001.

- [12] Gyula Simon, Miklós Maróti, Ákos Lédeczi, György Balogh, Branislav Kusy, András Nádas, Gábor Pap, János Sallai, and Ken Frampton. Sensor network-based countersniper system. pages 1–12, 2004.
- [13] Yi Shang, Wenjun Zeng, Dominic K. Ho, Dan Wang, Qia Wang, Yue Wang, Tiancheng Zhuang, Aleksandre Lobzhanidze, and Liyang Rui. Nest: Networked smartphones for target localization. *2012 IEEE Consumer Communications and Networking Conference, CCNC'2012*, pages 732–736, 2012.
- [14] Chunyi Peng, Guobin Shen, and Yongguang Zhang. BeepBeep: A High-Accuracy Acoustic-Based System for Ranging and Localization Using COTS Devices. *ACM Transactions on Embedded Computing Systems*, 11(1):1–29, 2012.
- [15] Fabian Hofflinger, Joachim Hoppe, Rui Zhang, Alexander Ens, Leonhard Reindl, Johannes Wendeberg, and Christian Schindelhauer. Acoustic indoor-localization system for smart phones. In *2014 IEEE 11th International Multi-Conference on Systems, Signals and Devices, SSD 2014*, 2014.
- [16] Vikas C. Raykar, Igor V. Kozintsev, and Rainer Lienhart. Position calibration of microphones and loudspeakers in distributed computing platforms. *IEEE Transactions on Speech and Audio Processing*, 13(1):70–83, 2005.
- [17] Marius H Hennecke and Gernot a Fink. TOWARDS ACOUSTIC SELF-LOCALIZATION OF AD HOC SMARTPHONE ARRAYS Intelligent Systems Group , Robotics Research Institute , TU Dortmund University , Germany. *Speech Communication*, pages 127–132, 2011.
- [18] Y Kuang, E Ask, S Burgess, and K Astrom. Understanding toa and tdoa network calibration using far field approximation as initial estimate. *ICPRAM 2012 - Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, 2:590–596, 2012.
- [19] K S Arun, T S Huang, and S D Blostein. Least-squares fitting of two 3-d point sets. *IEEE transactions on pattern analysis and machine intelligence*, 9(5):698–700, 1987.
- [20] Jorge J More. The Levenberg-Marquardt algorithm: Implementation and theory. *Lecture Notes in Mathematics*, 630(x):105–116, 1978.
- [21] Thomas F. Coleman and Yuying Li. An Interior Trust Region Approach for Nonlinear Minimization Subject to Bounds, 1996.
- [22] Simon Burgess, Yubin Kuang, Johannes Wendeberg, Kalle Å ström, and Christian Schindelhauer. Minimal solvers for unsynchronized TDOA sensor network calibration. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8243 LNCS:95–110, 2013.

- [23] David Freedman and Persi Diaconis. On the histogram as a density estimator: L^2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 57(4):453–476, 1981.
- [24] Doug Olenick. *Apple iOS And Google Android Smartphone Market Share Flattening: IDC*, 2015. <http://www.forbes.com/sites/dougolenick/2015/05/27/apple-ios-and-google-android-smartphone-market-share-flattening-idc/2/> [Accessed: Aug, 2015].
- [25] Wolfgang Mauerer, Gernot Hillier, Jan Sawallisch, Stefan Hönick, and Simon Oberthür. Real-Time Android: Deterministic Ease of use. *Proceedings of Embedded World Conference*, pages 1–7, 2012.
- [26] Igor Kalkov, Dominik Franke, John F Schommer, and Stefan Kowalewski. A real-time extension to the android platform. In *Proceedings of the 10th International Workshop on Java Technologies for Real-time and Embedded Systems*, pages 105–114. ACM, 2012.
- [27] Igor Kalkov, Alexandru Gurchian, and Stefan Kowalewski. Predictable broadcasting of parallel intents in real-time android. In *Proceedings of the 12th International Workshop on Java Technologies for Real-time and Embedded Systems*, page 57. ACM, 2014.
- [28] Mathias Obster, Igor Kalkov, and Stefan Kowalewski. Development and execution of plc programs on real-time capable mobile devices.
- [29] Ashraf Armoush, Dominik Franke, Igor Kalkov, and Stefan Kowalewski. An approach for using mobile devices in industrial safety-critical embedded systems. *Mobile Computing, Applications, and Services*, pages 294–297, 2014.
- [30] Yuan Cangzhou, Gao Chen, Dong Jibing, and Sun Wei. An optimizing scheme for wireless video transmission on android platform. In *Transportation, Mechanical, and Electrical Engineering (TMEE), 2011 International Conference on*, pages 970–973, Dec 2011.
- [31] Yin Yan, Shaun Cosgrove, Varun Anand, Amit Kulkarni, Sree Harsha Konduri, Steven Y Ko, and Lukasz Ziarek. Real-time android with rtdroid. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 273–286. ACM, 2014.
- [32] OAR Corporation. *RTEMS Real Time Operating System (RTOS)*, 2014. <http://www.rtems.org/> [Accessed: Mar, 2015].
- [33] Yin Yand, Shaun Cosgroved, Ethan Blanton, Steven Y Kod, and Lukasz Ziarekd. Real-time sensing on android. 2014.
- [34] David L. Mills. *Modelling and Analysis*, 2005.

- [35] Matthew Wright, Ryan J Cassidy, and Michael F Zbyszy. Audio and Gesture Latency Measurements on Linux and OSX Introduction and Prior Work. *Proceedings of the ICMC*, pages 423–429, 2004.
- [36] Eli Brandt and Roger Dannenberg. Low-Latency Music Software Using Off-The-Shelf Operating Systems. *Proceedings of the International Computer Music Conference 1998*, pages 137–141, 1998.
- [37] Adrian Freed, Amar Chaudhary, and Brian Davila. Operating systems latency measurement and analysis for sound synthesis and processing applications.
- [38] Viet-duc Le, Jacob Kamminga, Hans Scholten, and P J M Havinga. Measuring and Predicting Sensing Latency in Indeterministic Devices. 2015.
- [39] Google Inc. *Contributors to Audio Latency*, 2015. https://source.android.com/devices/audio/latency_contrib.html [Accessed: Mar, 2015].
- [40] Steven Rostedt. Finding Origins of Latencies Using Ftrace, 2009.
- [41] Google Inc. *Performance Tips*, 2015. <http://developer.android.com/training/articles/perf-tips.html> [Accessed: Mar, 2015].
- [42] Khronos Group. *The Standard for Embedded Audio Acceleration*, 2015. <https://www.khronos.org/opensles/> [Accessed: Mar, 2015].
- [43] Okan Turkes, Hans Scholten, and Paul J.M. Havinga. Blessed with opportunistic beacons: A lightweight data dissemination model for smart mobile ad-hoc networks. In *Proceedings of the 10th ACM MobiCom Workshop on Challenged Networks*, CHANTS '15, pages 25–30, New York, NY, USA, 2015. ACM.
- [44] Sameer Agarwal, Keir Mierle, and Others. *Ceres Solver*, 2015. <http://ceres-solver.org/> [Accessed: Aug, 2015].

Glossary

ADC Analog to Digital Converter. 40

API Application Programming Interface. 35

CLASS Cooperative Localisation on Android with ambient Sound Sources. 10, 11, 15, 17–19, 21–23, 31, 33, 38, 42, 45, 46, 48, 54, 55, 57, 58, 61–63

DSP Digital Signal Processor. 4, 33, 55, 62

FFA Far Field Approximation. 14, 18, 19, 63

FFT Fast Fourier Transformation. 4, 33, 41–43, 45, 55, 62, 63

FIFO First In First Out. 45

FIR Finite Impulse Response. 42

GPS Global Positioning System. 9, 19, 34–39, 47, 50–54, 63

HAL Hardware Abstraction Layer. 42

HBOS Histogram Based Outlier Score. 23, 27, 29, 31, 32, 58, 61, 63

HPET High Precision Event Timer. 14

HTTP Hypertext Transfer Protocol. 18, 46

MAC Media Access Control. 46

MLE Maximum Likelihood Estimation. 14

NDK Native Development Kit. 42

NTP Network Time Protocol. 34, 36–39, 48–54

OS Operating System. 4, 10, 11, 15, 17, 18, 31, 33–36, 41, 42, 45, 47, 48, 54, 55, 61

RANSAC Random Sample Consensus. 14, 27

RMSE Root Mean Square Error. 15, 19, 22, 57–59, 61

RT Real-Time. 34, 61

SNR Signal to Noise Ratio. 18, 20, 21, 44, 62

SQL Structured Query Language. 21, 37, 46

TDOA Time Difference Of Arrival. 10, 11, 13, 14, 17–19, 23–28, 31, 33, 46–55, 58, 61–63

TOA Time Of Arrival. 9, 10, 12, 15, 21, 24, 27, 28, 31, 46, 63

WSN Wireless Sensor Network. 12