Scalable positioning of commodity mobile devices using audio signals

---

A dissertation submitted towards the degree Doctor of Engineering of the Faculty of Mathematics and Computer Science of Saarland University

by

Viktor Tamás Erdélyi

Saarbrücken

October, 2018

Date of the colloquium:	February 14th, 2019
Dean of the faculty:	Prof. Dr. Sebastian Hack
Chairman of the Examination board:	Prof. Dr. Markus Bläser
1st reviewer (supervisor):	Prof. Dr. Peter Druschel
2nd reviewer:	Dr. Deepak Garg
3rd reviewer:	Dr. Akira Uchiyama
Academic Assistant:	Dr. Engel Lefaucheux

Kurzdarstellung

Diese Dissertation befasst sich mit dem Problem, eine Positionskarte von sich am gleichen Ort befindenden mobilen Geräten zu berechnen. Dies soll skalierbar, ohne Verwendung von spezialisierter Hardware oder Infrastruktur (ausgenommen einfache WLAN- oder Mobilfunkzugang) erfolgen. Bei Veranstaltungen wie Meetings, Diskussionen oder Konferenzen kann eine Positionskarte die Benutzer bei spontaner Kommunikation mithilfe der relativen Positionen in zweierlei Hinsicht unterstützen. Erstens ermöglicht sie den Benutzern, die Empfänger von Nachrichten aufgrund deren Position zu wählen, was auch eine positionsabhängige Verteilung von Unterlagen erlaubt. Zweitens ermöglicht sie den Sendern, ihre Position in die Nachrichten zu integrieren, was eine Interaktion zwischen Referent und Zuhörer in einem Hörsaal und die Sammlung von positionsabhängigen Rückmeldungen erlaubt.

In dieser Dissertation stellen wir die Mobile-App und das System Sonoloc vor, das mithilfe von Tonsignalen erlaubt, die *relative* Position handelsüblicher, intelligenter Geräte zu bestimmen. Sonoloc kann eine beliebige Zahl von Geräten innerhalb des Hörbereichs durch eine gleichbleibende Zahl von Tonsignalen, die von einer Teilmenge der Geräte gesendet werden, lokalisieren. Unsere experimentelle Analyse mit bis zu 115 Geräten in echten Räumen zeigt, dass das System trotz signifikanter Hintergrundgeräusche unter Verwendung von bis zu 15 Tonsignalen mit einer Genauigkeit von wenigen Dezimetern Geräte lokalisieren kann.

Abstract

This thesis explores the problem of computing a position map for co-located mobile devices. The positioning should happen in a scalable manner without requiring specialized hardware and without requiring specialized infrastructure (except basic Wi-Fi or cellular access). At events like meetings, talks, or conferences, a position map can aid spontaneous communication among users based on their relative position in two ways. First, it enables users to choose message recipients based on their relative position, which also enables the position-based distribution of documents. Second, it enables senders to attach their position to messages, which can facilitate interaction between speaker and audience in a lecture hall and enables the collection of feedback based on users' location.

In this thesis, we present Sonoloc, a mobile app and system that, by relying on acoustic signals, allows a set of commodity smart devices to determine their *relative* positions. Sonoloc can position any number of devices within acoustic range with a constant number of acoustic signals emitted by a subset of devices. Our experimental evaluation with up to 115 devices in real rooms shows that – despite substantial background noise – the system can locate devices with an accuracy of tens of centimeters using no more than 15 acoustic signals.

Contents

Chapter 1

Introduction

In this thesis, we set out to produce a *relative position map* for mobile devices. A *relative* position map is a map of a set of devices where each device is associated with a coordinate describing its position relative to other nearby devices. These coordinates are relative in the sense that they are not anchored to any global coordinate system. The map should be computed without requiring specialized hardware or specialized infrastructure in a way that scales to hundreds of devices, and the map should be accurate in the sense that the worst-case positioning error should be low enough to distinguish between neighboring seats.

Assuming that users provide (at least temporary) contact information, a relative position map enables users to be addressed based on their position relative to others. This ability can provide context for communication in two ways: (1) by addressing communication to receivers based on their position relative to the sender, and (2) by tagging communication with the position of the sender relative to the receiver. In the following, we present a number of example applications for each of these categories.

Addressing receivers based on relative position

When a user wants to send files and/or messages to one or more recipients, they may want to select the set of recipients based on a relative position map. The content to be sent could depend on the position of the recipient.

In some cases, the ID of the recipient is important for the sender. For instance, in events such as conferences, meetings, or other informal events, a user can use a relative position map to send follow-up messages to specific people whose relative position they can remember. As another example, in meetings or dining events with new acquaintances, in order to help the user get to know whom they are interacting with, an app could show each participant's position at the table along with their name.

There are also cases where it is not important to know the ID of the recipient; instead, only their position (e.g., in a specific seat, at the same table, or, in a given area of the room) is important:

- Event organizers may wish to disseminate information or tasks to attendees based on their position in the room (e.g., at a given table) rather than based on their identity.

- When an instructor distributes exams to students, he or she may prefer to make sure that, regardless of the students' identity, all neighboring pairs of students get different exams. Additionally, when examining cheating incidents, it is useful to know the ID *and* position of each exam taker. Such information can be automatically generated using a relative position map, eliminating the need to record manually where people are sitting.

- In concert halls and sports venues, the relative position map could be used to turn the area occupied by the audience into a large screen on which the organizers could display words, images or even animations. Based on the participants' position, the organizers could automatically assign a color to each participant, effectively using each device as one of the "pixels" of the large screen. The color would be displayed either by the participants' smartphone screens, or by hand-held LED lights handed out at the entrance.[1]

- Assuming that attendees of a concert-like event create a relative position map before the event starts, in an emergency where the venue needs to be evacuated, a server could decide the (globally) optimal escape route for all devices and display the position-specific escape route on each attendee's device.

Tagging communication with the sender's relative position

The use cases discussed above involved sending information to users at a given position. There are also use cases in which the important information is the position of the *sender* of the message (as opposed to that of its recipient):

- Consider a lecture in a large auditorium where an audience member submits a question or comment electronically (either with or without an ID attached). In this case, a position map would make it possible to indicate the sender's position on the speaker's monitor, making it easier for the speaker to make eye contact with the sender.

- During an exam, a student sends a message to the lecturer, requesting the lecturer's presence at the student's seat in order to answer the student's question. Again, the lecturer would need an indication of the sender's position.

- Attendees of a larger meeting or conference talk may wish to notify the organizers unobtrusively (by a few button presses on their phone, without interrupting the flow of the event) that they want to ask a question and they need a microphone, or that their headset stopped working. Attaching the position of the source to the message can enable such interactions.

[1] Either the hand-held LED light must have built-in relative positioning support or it must be paired with the user's smartphone in advance.

- If, in a crowded concert hall, a person requires medical assistance, people nearby may send an emergency message to the organizers. If the location of the emergency could automatically be attached to the message, the organizers may be able to respond to the situation faster.

- Organizers of events in classrooms, concert venues, or sports arenas may want to collect, unobtrusively, feedback about the audiovisual experience based on the seat or section a user occupies. Although, in some cases, users could easily attach their seat number to their feedback, in venues where participants can choose their positions prior to the event (e.g., a venue where seats can be moved, or one without seats), it may be more effective to attach their position automatically to their feedback.

- People who take photos/videos of such an event or performance could upload it to a specific website with their position attached. The website could then show the position map, and, for each device, show either thumbnails or the number of photos taken by the device's owner.

- The participating devices could form an ad-hoc sensor network. Each device is equipped with a given set of sensors and knows its position relative to the others, so they can attach position information to each sensor reading they send. This capability enables position-based monitoring of sound pressure level, air quality, temperature, humidity, or anything else that the phone can sense.

Clearly, there are a number of uses of relative position maps. In the following, we discuss ways to produce such maps.

1.1 Producing relative position maps

One approach to produce a relative position map is to equip the room with some sort of infrastructure. The infrastructure nodes first exchange signals with the mobile devices. Then, these devices are localized relative to the infrastructure nodes, based on fingerprint maps, angle of arrival, distance measurements or time difference of arrival measurements. These systems are called infrastructure-based systems or indoor localization systems. (For a recent survey of indoor localization systems, see [56].)

To maximize deployability, it is desirable to require only commonly available infrastructure such as basic cellular coverage and basic Wi-Fi access [6] via standard, off-the-shelf access points. However, in order to position devices relative to the infrastructure nodes, additional information is needed. A common approach is to measure a rough distance from each access point based on received signal strength (RSS) measurements, which requires prior knowledge about the positions and transmission power settings of the access points. An alternative approach is to create an RSS fingerprint map, which involves measuring RSS from all access points at each

location of interest and then manually attaching a position to each such measurement. However, the RSS-based approach is not accurate enough to discern neighboring seats reliably, and they are sensitive to changes in the environment, which affect the RSS. Additionally, they require explicit pre-deployment effort.

Compared to approaches that rely on commonly available infrastructure (such as RSS-based approaches), a more accurate alternative would be to install customized anchor nodes. However, due to their significant pre-deployment effort, such systems are not widely used today. As the hardware required for the anchors becomes cheaper, the deployment barrier may gradually become lower, but, in the foreseeable future, there will likely be some locations where the necessary infrastructure does not exist (e.g., outdoors), does not have a high enough density, or – for example due to a natural disaster – is out of service. Therefore, it is preferable to use an approach that can work without *specialized infrastructure* – that is, only using commonly available infrastructure such as basic cellular coverage and basic Wi-Fi access without explicit pre-deployment effort or the need for installing anchor nodes.

An alternative approach that eliminates the need for infrastructure nodes is to rely only on device-to-device (D2D) communication for localization. In the D2D approach, due to the lack of infrastructure nodes with known positions, the position of devices can only be determined relative to each other. Existing D2D approaches rely on some form of time of flight measurement and then use the known signal propagation speed to convert the times of flight to distance estimates, and produce a position map based on the resulting distance constraints.

Such systems use either electromagnetic waves known as radio signals or acoustic signals. Radio signals are unobtrusive and they offer high bandwidth – hundreds of MHz of bandwidth in ultra-wideband (UWB) systems. High-bandwidth signals allow fast communication and have desirable signal analysis properties. These advantages make radio signals attractive, but – in order to achieve good accuracy – communication endpoints need access to the individual samples being transmitted and received. For radio waves, however, such access is not available on commodity devices, so specialized hardware is required. Thus, for D2D positioning on commodity devices, radio signals are not feasible, and we need to use acoustic signals.

Regarding audio signals, we have three options: high-frequency ultrasound (hundreds of kHz or even MHz-range), low-frequency ultrasound (20 kHz–24 kHz at 48 kHz sampling rate) and audible signals (0 kHz–20 kHz). High-frequency ultrasound would offer high bandwidth, but it is not feasible for two reasons: (1) the strong attenuation of such high-frequency sounds severely limits their communication range, and (2) commodity mobile devices do not support such signals. Low-frequency ultrasound would minimize user annoyance, but, unfortunately, the audio hardware on current commodity devices is not able to transmit with high enough power and receive with sufficient sensitivity at ultrasonic frequencies. Therefore, in order to support commodity devices, we need to use audible signals.

However, existing acoustic D2D approaches do not scale to hundreds of devices. Prior systems typically rely on two-way ranging for accurate distance estimation. For a given pair of

devices, two-way ranging requires a two-way signal transmission from both devices, as shown in BeepBeep [65]. Extending two-way ranging to support n devices might seem to require n^2 transmissions, but this is not the case. If all devices transmit one signal each (a total of n transmissions), then each pair of devices will have the required two signals to perform two-way ranging. However, in situations such as large lecture halls, transmitting one signal per device would still require hundreds of audio signals, which would be impractical because (1) in the case of audible signals, they would make too much noise, (2) it would take too much time to transmit them all (in our experience, at least a few hundred milliseconds per signal).

In this thesis, we show how to localize all devices using a significantly reduced number of audio transmissions. We show that it is sufficient to transmit audio signals from a small subset of devices (called transmitters), and the remaining devices (called passive devices) only need to listen to all transmissions passively. The key is to position the devices in two phases using two different methods. In the first phase, the transmitters use BeepBeep [65] to estimate all pairwise distances among each other based on differences between the arrival times of signal pairs, which provides enough constraints to compute their 2D positions relative to each other. In the second phase, we localize the passive devices. For these devices, we cannot obtain pairwise distance information. Instead, for each passive device P, we estimate a set of *distance differences* from P to pairs of transmitters. Based on these distance differences, the passive devices compute their 2D positions relative to each of the transmitters.

In order to localize devices with a small number of transmitters *with good accuracy*, the set of transmitters must be carefully chosen. Towards this end, we propose an iterative protocol that proceeds in several rounds. Initially, a set of randomly selected devices emit a sequence of audio signals while simultaneously listening. Using the two-phase positioning method described above, the devices can tentatively place themselves on a map. Then, in each subsequent round, an additional transmitter is selected strategically based on the latest intermediate position map, which emits another signal in order to refine the map in the next round. This iterative protocol enables us to have an accurate relative position map.

1.2 Contributions

The thesis makes the following contributions:

- We develop two new techniques:
    1. A two-phase device localization protocol that is scalable enough to position any number of devices using signal transmissions from only a small, constant number of participating devices. This protocol improves over naïve ranging approaches, which would require $\Theta(n)$ transmissions to position n devices.
    2. An iterative transmitter selection algorithm in order to retain good accuracy despite the small number of signal transmissions.

- We design and present Sonoloc, a system that, upon the request of one user, can compute a relative position map of hundreds of commodity smart devices within acoustic range using audio signals. Sonoloc computes the position map in a scalable and accurate way without requiring specialized infrastructure. Our prototype relies on Internet access, but, in principle, Sonoloc could also operate without any infrastructure at all, instead relying only on device-to-device network communication.

- We present results of extensive simulations with many device layouts as well as an experimental evaluation of a Sonoloc prototype with up to 15 Motorola Nexus 6 smartphones and 100 Raspberry Pi-based microphone-only devices in several different real rooms and with different background noise levels. We believe that this evaluation goes far beyond prior work. Our results show that Sonoloc can position hundreds of devices with an accuracy of tens of centimeters with no more than 15 audio chirps, and with background noise up to 62 dB.

Chapter 2

Related Work

In this section, we discuss prior work on localization. Localization methods can either rely on infrastructure (Section 2.1) or device-to-device measurements (Section 2.2). Table 2.1 presents a comparison of representative localization protocols, and shows that each existing system has at least one of the following limitations:

- Is not scalable (in particular, in the case of acoustic signals, they require N beeps for N devices) [65, 29, 52]
- Requires specialized infrastructure [85, 83, 84]
- Requires specialized hardware [89, 85, 83, 84]
- Is not accurate enough to place people into a particular seat [16, 84, 22]

In contrast, applications we target require that the relative positioning of a potentially large number of people (more accurately, devices carried by them) be preserved, and that these applications be deployed on commodity devices, without specialized infrastructure support.

2.1 Infrastructure-based localization

To perform localization, infrastructure-based localization protocols rely on existing infrastructure such as Wi-Fi access points or speakers with known positions. These techniques are typically used indoors, where GPS [60] is often unavailable, and they are also known as indoor localization. They localize devices relative to the infrastructure they rely on. Since the geographic positions of such infrastructure devices are typically known, these techniques effectively produce absolute geographic coordinates.

Infrastructure-based systems can be based on fingerprint maps, angle of arrival estimations, time of flight estimations, or time difference of arrival (TDOA) estimations. Additionally, some hybrid systems combine multiple approaches.

Protocol	Medium	Accuracy	Max. error	Tested range	Custom hardware	Scalable?	Meas. speed	Spec. infra.	Notes
PinPoint [89]	RF	>1 m avg.	4.3 m	44.5 m	All	yes ($\Theta(n)$)	fast	no	custom RF HW for fine-grained timestamps
ArrayTrack [83]	RF	0.26 m med.	–	floor	All	yes	fast	yes	custom APs with 6+ antennas, custom client HW
Synchronicity [84]	RF	1.6 m med.	–	floor	APs	yes	fast	yes	Requires SW/HW change at APs
ToneTrack [85]	RF	0.9 m med.	5 m	floor	All	yes	fast	yes	Custom devices and APs
Horus [88]	RF	0.5 m med.	2.9 m	floor	-	yes	fast	yes	Wi-Fi fingerprint-based
EZ Localization [16]	RF	2 m–6 m med.	20 m	floor	-	yes	fast	no	Integrates GPS, server support
SpotFi [38]	RF	0.4 m med.	12 m	floor	-	yes	fast	yes	AP SW change
Ubicarse [39]	RF	0.4 m med.	–	floor	-	yes	fast	yes	Requires known AP positions; localizes non-RF objects via vision
Locatify (UWB) [54]	RF UWB	< 0.2 m	–	room	All	yes	fast	yes	TDOA-based using beacons, mobile node transmits
BeepBeep [65]	AR	0.02 m avg.	–	12 m	-	no	slow	no	pairwise distance only, line-of-sight results
Ping-Pong [29]	AR	0.03 m–0.2 m	–	2.8 m	-	no	slow	no	Error device-size dependent
FAR [91]	AR	0.02 m med.	–	<2 m	-	no	fast	no	pairwise distance only, 12Hz updates
Liu et al. [52]	AR	0.1 m–0.2 m med.	0.35 m	7.3 m	Beacons	no	slow	yes	Requires deploying anchor nodes. Limited eval. with 1 phone in 2 locations.
GuoGuo [51]	AR	0.06 m–0.25 m avg.	–	>15 m	Beacons	yes	medium	yes	TDOA-based multilateration, >1Hz updates
Centaur [59]	RF+AR	0.6 m med.	3.5 m	floor	-	no	slow	yes	Combines Wi-Fi and acoustic
ALPS [41]	RF+AR	0.1 m–0.2 m med.	3.2 m	72 m^2	Beacons	yes	fast	yes	Requires ultrasound HW beacons
Akiyama et al. [4]	AR+Light	0.01 m	–	<5 m	Beacons	yes	fast	yes	Requires sync. LED beacons
SmartSLAM [22]	RF+Inert.	1.6 m (66%), 2.7 m (95%)	–	floor	-	yes	medium	partial	2Hz updates, sensor fusion (PDR, Wi-Fi fingerprinting, magnetic)
Sonoloc	AR	0.06 m med.	0.5 m	17.8 m	-	yes	slow	no	Position map with up to 15 signals

Table 2.1: Survey of representative positioning and location protocols using radio waves (known as radio frequency or RF) or audio signals (known as acoustic ranging or AR). "Fast" measurements require less than 100 ms, whereas "slow" ones require more than 1 s. In the accuracy column, we abbreviate "average" as "avg.", and "median" as "med.".

2.1.1 Fingerprint maps

The fingerprint map approach typically uses Wi-Fi and relies on access points (APs) for positioning. Client devices create a fingerprint map by measuring received signal strength (RSS) from different APs at many points within indoor locations, and use such maps to determine their location probabilistically. An example of a fingerprint map-based system is presented in [88].

The downside of fingerprint map-based approaches is their sensitivity to changes in the environment, which make previous signal strength measurements outdated, leading to worse accuracy. Changes that affect the observed signal strength (and, indirectly, positioning accuracy) include:

- Changes in the APs' position, density, configuration (e.g., transmission power) or hardware
- Changes in the reverberation properties of the room, due to changes in the position and density of people or objects (such as furniture) in the indoor space

In order to regain accuracy, the fingerprint map needs to be re-computed.

2.1.2 Angle of arrival

The angle of arrival approach relies on multiple antennas installed in each wireless AP to observe phase difference of arrival or time difference of arrival information between pairs of antennas, and use it to estimate the angle of arrival of an incoming wireless signal. While the approach can achieve sub-meter resolution [83, 85], due to its need to access the physical layer wireless signal, it requires specialized AP hardware. The positioning accuracy varies depending on the density and construction of base stations (number of stations and antennas per station).

2.1.3 Distance-based systems

Distance-based systems rely on anchors with known positions. They estimate pairwise distances between a mobile node and all anchors using one of several available techniques: based on RSS measurements, based on two-way ranging, based on direct time-of-flight measurements or using the phase accordance method. Time of flight information can be turned into distance information via multiplication by the known signal propagation speed. Once distance estimates are available, they can be used as distance constraints to determine the location of the mobile node relative to the anchors using some form of trilateration technique.

The RSS-based estimation typically uses one-way transmissions from anchors to mobile nodes, and uses Bluetooth Low Energy beacons as anchors (such as earlier versions of the Locatify system [11]), but other combinations such as mobile-to-anchor one-way transmission or using Wi-Fi RSS are also possible. The key idea is to construct a model that maps RSS values to distances based on either empirical training data or based on a signal propagation model. The

downside of RSS-based approaches is that RSS is strongly influenced by environmental factors such as multipath, and thus they have low accuracy.

Two-way ranging systems rely on the idea of BeepBeep [65]. BeepBeep is a protocol that estimates the distance between two devices by serially emitting an acoustic chirp on both devices' speaker, listening on the microphones of both devices, and computing the distance based on differences in the observed signal arrival times. BeepBeep's distance estimation accuracy is around 1 cm–4 cm median error. For more details on distance estimation with two-way ranging, see Section 6.1. To perform two-way ranging, participating nodes must be able to access the raw stream of samples transmitted and received. When using acoustic signals, the I/O API available on standard smartphones provides such access. However, in the case of RF signals, standard APIs only offer higher-level functionality such as scanning for base stations or sending/receiving 802.11 data packets, and they do not provide access to the raw samples. Therefore, in order to perform two-way ranging with RF signals, specialized transceiver hardware (such as a software-defined radio) is required. For instance, Locatify's earlier ultra-wideband system [54] used this approach.

The third way to estimate distance between an anchor and a mobile node is to perform direct time-of-flight measurements and convert them to distance using the known propagation speed of the signal. There are two approaches to estimate time of flight. The first approach is to compare the transmitter's transmission timestamp to the receiver's reception timestamp directly, which requires the clocks of the transmitter and the receiver to be synchronized. Additionally, at least on commodity smartphone hardware, there are unpredictable delays in the I/O stack on both the transmitter and the receiver, which makes it difficult to perform the fine-grained timestamping required for accurate time-of-flight estimation. The second, more indirect approach is to send a radio signal *and* an audio signal at the same time, and take advantage of the large difference in their propagation time. Then, since the radio signal travels at the speed of light, its the time of flight is negligible. Thus, at the receiver, the difference between the two signal arrival times gives an approximation of the time of flight of the audio signal. This approach is used by Cricket [66], though the resulting distance is used only to identify the closest anchor. Again, due to requiring access to the individual signal samples, the approach is not feasible on commodity smartphones.

The phase accordance method [28] uses a pair of sinusoidal waves with slightly different frequencies and observes the phase difference of the two waves to estimate the distance between a transmitter and a pair of co-located receivers. A notable system that uses this method is SyncSync by Akiyama et al. [3, 4], which achieves high accuracy (6 cm median error) by using loudspeakers as anchors and modulated LED light captured by the smartphone camera for synchronization. SyncSync extends the original phase accordance method with frequency division multiplexing to enable multiple loudspeakers to be used as anchors. Due to its reliance on single-frequency sinusoidal carrier waves and the ability to extract phase information from them, it is unclear to what extent the system can withstand background noise.

2.1.4 TDOA-based systems

Time difference of arrival (TDOA)-based systems estimate time difference of arrival for triples of devices. TDOA can be defined given a triple of either 2 transmitters and 1 receiver, or 2 receivers and 1 transmitter. In both cases, estimated TDOA values can be turned into distance differences by multiplying them by the signal propagation speed: for a triple of devices (A, B, C), the distance difference refers to the difference between the A-to-C distance and the B-to-C distance), which can be used to determine the position of a mobile node relative to the anchors. Although distance difference information is more implicit than distance information and generally has lower positioning accuracy, it is useful when pairwise distance information involving the mobile node is not available.

Similar to time of flight-based systems, systems based on TDOA also rely on pre-installed anchors with known positions. However, in order to determine TDOA, the anchors must additionally be synchronized to each other.

There are two main approaches to build a TDOA-based system. One approach is to transmit signals from the anchors and determine their TDOA at the mobile node, which leads to a localization problem called "sensor localization based on sources with known positions". The other approach is to transmit a signal from the mobile node and determine its TDOA at the anchors, which is known as "time-difference-of-arrival (TDOA)-based source localization", or, if acoustic signals are used, as "acoustic source localization using a synchronized microphone array". Due to the symmetry of the problem, these two types of localization problems are equivalent and can be solved by the same methods. Closed-form solutions include the *spherical interpolation method* [76, 79, 80], the *hyperbolic intersection method* [15, 86], and the *linear intersection method* [12]. A survey of existing solutions is presented in Stoica et al. [80]. Sonoloc uses a 2D variant of the spherical interpolation method [80] to localize passive devices (see Section 6.2.2).

Similarly to two-way ranging, the estimation of TDOA requires access to the raw samples transmitted and received. Thus, using RF signals implies the need for specialized hardware. At the time of writing, the most recent implementation of Locatify [54] takes this approach using ultra-wideband radio signals where users need to carry around a special transmitter device. A common workaround to avoid the need for special hardware is to rely on audio signals. For instance, an indoor location system by Lazik et al. [40, 41] achieves 15 cm–50 cm accuracy based on supersonic chirps (that minimize human audible artifacts); Liu et al. [52] obtain 20 cm–40 cm median positioning error. Akkurate [55] and Aguilera et al. [2] achieve 10 cm accuracy in 95% and 90% of the cases, respectively.

2.1.5 Hybrid approaches

There are also systems that combine multiple techniques and/or data from multiple sensors (e.g., by incorporating inertial sensor data). For instance, Centaur [59] uses the distances measured

using AR as geometric constraints for Bayesian inference on the RF profiles of the participating devices. Centaur also introduces a new AR measurement technique, DeafBeep, which works with speaker-only devices and can be directly integrated with Sonoloc. In Liu et al. [50], BeepBeep-based AR ranging informs Wi-Fi profile-based localization. CUPID [77] combines angle-of-arrival-based and distance-based methods, and achieves a median localization error of 2.7 m. Finally, given an initial location (by Wi-Fi or GPS), SmartSLAM [22] performs simultaneous localization and mapping (SLAM) using Wi-Fi RSSI-based fingerprinting, inertial sensors (accelerometer, gyroscope) and compass on smartphones. In order to minimize CPU load and energy consumption, the system moves between different sensor fusion algorithms depending on available prior knowledge of both user and environmental parameters; its localization error is several meters. The accuracy of all these systems is still insufficient for our purposes, and their RSS-based Wi-Fi components are sensitive to changes in the environment.

In contrast to all infrastructure-based systems, Sonoloc is able to determine relative positions of devices accurately and without relying on specialized infrastructure.

2.2 Positioning based on device-to-device measurements

In contrast to infrastructure-based approaches, devices can be positioned without infrastructure support, instead based solely on peer measurements. Doing infrastructure-free localization involves shifting the functionality of the infrastructure to one or more mobile nodes. Since there are no anchors to rely on, the coordinates produced by the localization are also relative and are not anchored to any global coordinate system.

The absence of infrastructure also limits the localization techniques we can use. For instance, fingerprinting cannot be used because it relies on the presence of APs with a certain density. Using a radio-based angle-of-arrival method would require that the mobile nodes have access to the individual samples of the wireless signal on all antennas, which, due to the lack of API support, would require specialized hardware.

Another option would be an *acoustic* angle-of-arrival approach, which would require access to the recorded samples on more than 2 microphones (typical smartphones have only 2), and devices' orientation would need to be tracked. Even then, accuracy would be a concern. For a simple back-of-the-envelope calculation, assume 15 cm distance between two microphones (the size of a typical smartphone). Then, their maximum possible time difference of arrival would be 0.44 ms, and this could be both positive and negative, so the total range of TDOA values would be 0.87 ms. At 48 kHz, this range can be divided into ca. 42 samples. The system has to assign an estimated angle to all 42 possible TDOA values. Given a 360° angle range, this leads to an angle resolution of 8.6° — *assuming perfect signal analysis*. Each additional 1-sample error in signal analysis would increase the angle error by another 8.6°. As a reference, we found that 5–10 samples of error are common in signal analysis, which would lead to a total angle error of 42.9°–85.8°. We believe that this is accuracy is insufficient for our purposes.

TDOA-based methods cannot be applied directly because of the absence of anchors with known locations. Still, joint localization of many devices based on only TDOA information is possible: closed-form and near-closed-form solutions for this problem are presented by Le et al. [45, 47, 42]. However, because of the synchronization requirement of TDOA estimation, these solutions are difficult to deploy on commodity devices. (It is possible to perform such synchronization by transmitting calibration signals among devices with both transmission and reception capability; see Chapter 6 for details.)

Thus, what remains as the most promising approach is the distance-based approach. Similarly to TDOA, the absence of infrastructure complicates things. Instead of positioning one device relative to the anchors with known positions, we now need to position *all* devices simultaneously. In order to produce a relative position map, first we need to collect pairwise distances among all devices. Then, we need to produce coordinates for each device in an appropriately dimensioned space in a way that the coordinates satisfy the pairwise distance constraints. A well-known way of producing coordinates based on distances is multidimensional scaling (MDS) [87, 82, 34]. A different approach is used in Virtual Compass [10]. Virtual Compass obtains pairwise distances from Bluetooth- and Wi-Fi-based RSS measurements, and computes a 2D relative position map using the Vivaldi algorithm [19], which calculates force vectors based on pairwise distances, and iteratively refines the nodes' coordinates by moving them along the resulting forces. Due to its reliance on RSS measurements, the accuracy of Virtual Compass is limited. WASP [75] uses radio signals and specialized hardware to provide localization for sensor networks, and estimates distances based on two-way ranging. Other approaches use acoustic ranging to obtain pairwise distances; Qiu et al. [72] (where the position map is used for file sharing), Ping-Pong [29] (where each measurement emits a musical note) and Ayllón et al. [8] (a combination of AR and inertial sensor readings).

2.2.1 Distance-based joint source and sensor localization

The above pairwise-distance-based techniques do not distinguish between the speaker and microphone of a given device, and instead assume that devices are point-like, i.e., the speaker and the microphone of a device are at the exact same position. However, if this assumption does not hold, these methods will not work well. In contrast, distance-based joint source and sensor localization considers sources (speakers) and sensors (microphones) as separate entities and localizes them separately, without assuming any knowledge of their positions. This localization problem was studied previously in [64, 44, 43, 47]. For each (speaker, microphone) pair, their distance needs to be provided as input, and the technique outputs relative coordinates for all speakers and microphones.

Since humans live in a 3D space, ideally, we want to obtain 3D coordinates for each node. Towards this end, Le et al. [44, 43] presented closed-form and near closed-form solutions in 3D space.

Closed-form solutions in 3D space: Given distances for all speaker-microphone pairs, closed-form solutions were presented in [44] for coordinates of speakers and microphones. Real experiments presented in [46] show that if the number of speakers and microphones is large, the closed-form solutions proposed by [44] give very accurate estimations. For example, if we set up a dozen speakers and 121 microphones in a room of dimensions $8.15\,\text{m} \times 3.9\,\text{m} \times 3.35\,\text{m}$, with a reverberation time of $T_{60} \simeq 0.7\,\text{s}$ and the minimum and maximum distances between speakers and microphones are around 0.2 m and 6 m, respectively, then the predicted root mean square error (RMSE) of the estimations is smaller than 4 cm.

Near-closed-form solutions in 3D space: Based on the closed-form solutions of [44], in [43], Le et al. developed near closed-form solutions that work well even for a small number of speakers and microphones. Near closed-form solutions are closed-form functions with one or two unknown parameters. These unknown parameters can be determined by grid search, i.e., by an exhaustive search through a manually specified subset of the parameter space. In cases where the input distances are noisy, the near closed-form solutions are always better than closed-form solutions. However, due to the use of grid search, the execution time of near closed-form solutions is much higher (around 500–1000 times higher) than that of closed-form solutions.

Solutions for 3D space can produce 3D coordinates, which makes them attractive, but they also have a number of disadvantages compared to 2D variants: (1) they need a higher number of transmitters, (2) their computation time is higher, and (3) the transmitters need to be spatially distributed along all three axes of the 3D space.

Closed-form solutions in 2D space: In many realistic situations, however, all devices almost fit into a 2D plane. For instance, in a meeting room where all participants have their devices on the table, all devices will fit into a horizontal 2D plane. In other situations, such as in a lecture hall where different rows are located at different heights, the plane may be slightly tilted. When devices almost fit into a 2D plane, it is not feasible to distribute transmitters across the axis perpendicular to the 2D plane of devices, thus a 2D localization is more appropriate. Towards this end, based on the results in [44, 43], Section 3.2 of [47] proposed closed-form solutions for 2D space.

A common downside of all pairwise-distance-based methods is the lack of scalability. Since a pairwise distance can be determined only when both sides of the pair transmit a signal, in order to position n devices, $\Theta(n)$ transmissions are required. Since we need to use audible signals, this approach does not scale to large venues with hundreds of devices for the following reasons. First, since transmissions are audible, hundreds of transmissions would become very disruptive to human users. Second, transmissions need to be sufficiently separated in time or spectrum for good signal-to-noise ratio. We briefly mention frequency division and code division multiplexing in Section 8.3. When using time-based separation (time division multiplexing), since each transmission takes hundreds of milliseconds, hundreds of transmissions would lead to very large delays.

Sonoloc uses the 2D method from [47] to localize *transmitters* (see Section 6.2.1). In

order to scale to a large number of devices, Sonoloc uses only a constant number of transmitters *regardless of the number of devices in the room*. The non-transmitting devices within audio range record the transmitted signals, and estimate distance differences in order to localize themselves relative to the transmitters. As Table 2.1 shows, compared to prior AR-based systems that do not require specialized infrastructure, Sonoloc achieves similar median accuracy. However, in contrast to prior work, which either did not evaluate maximum error, have several meters of maximum error, or have limited-scope evaluation, Sonoloc has maximum errors of around 0.5 m. This is required to position users relative to each other correctly in practice.

2.2.2 High-speed tracking systems

Finally, some systems focus on continuously maintaining a real-time estimate of distance or relative position, but only work for a single pair of phones. For instance, the FAR system [91], which was designed for mobile gaming, uses AR to continuously track the distance between two phones with a median error of 2 cm. Measurements are quick (< 100 ms) but are pairwise only. Another example is the protocol in Qiu et al. [71], which maintains the relative 3D position of (only) two phones using AR cues, inertial sensors, two microphones per phone, and a digital compass. In contrast to these systems, Sonoloc performs a one-time positioning of *many* devices with a small, constant number of audio transmissions.

Chapter 3

System design

In this chapter, we describe Sonoloc's overall design and its iterative protocol for computing a relative position map among the set of participating devices.

In order for a relative positioning system to be practical, it needs to be accurate, scalable, and it needs to work in real environments (e.g., with background noise) without specialized infrastructure and without specialized hardware. The key challenge is to produce a relative position map under the above constraints. For instance, in order to make the design scalable and minimize its intrusiveness, it is important to ensure that, unlike existing solutions based on two-way ranging, not all users' devices have to emit sounds. The lowered number audio transmitter devices (*transmitters*) introduces another challenge: the problem of selecting the set of transmitters.

Since Sonoloc relies on acoustic ranging to estimate distances and distance differences among devices, it needs to be able to detect its audio signals accurately in noisy environments, and determine the difference between the arrivals of any two signals. The signal design and processing used by Sonoloc for this purpose are described in Chapter 5. In this chapter, we assume that acoustic signals can be detected and the required arrival time differences can be obtained.

Given a set of transmitters and the signal analysis results for all signals at all devices, the computation of a relative position map involves the following steps:

1. **Distance estimation:** Given the signal analysis results, estimate all pairwise distances among the transmitters.
2. **Distance difference estimation:** Given the signal analysis results, for each *passive device* (i.e., a device not chosen as a transmitter), estimate the distance difference between that device and each pair of transmitters.
3. **Positioning of transmitters:** Given estimated pairwise distances among transmitters, compute a two-dimensional map of transmitters that is consistent with the pairwise distances.
4. **Positioning of passive devices:** Given estimated distance differences, place the pas-

sive devices into the same map so that their positions are consistent with the distance differences.

The details of the techniques we use to perform these steps are described in Chapter 6. In this chapter, we assume that a position map can be computed from the output of the signal analysis.

3.1 Assumptions

Sonoloc makes the following assumptions:

- Participating users carry a smart device that has support for Bluetooth Low Energy [78], a Wi-Fi [6] or cellular network connection, a microphone and a speaker. All modern smart phones have these capabilities.
- The distance between the speaker(s) and microphone(s) for each phone is known. For this purpose, the app provider can run a simple online database with the measured distances for all common device models.
- All participating devices are within audible distance from each other. Since devices are located in spaces at most as large as a lecture hall in Sonoloc's intended applications, this is a reasonable assumption. In addition, our experimental evaluation confirms that devices located at opposite ends of a lecture hall can still hear each other.
- Users place their phones into a relatively unobstructed position (e.g., by holding them up in the air or by placing them on the table).
- Phones do not move while Sonoloc performs measurements. This assumption is consistent with Sonoloc's usage, where a user initiates the system when participants are seated in an auditorium or at a table.
- Because Sonoloc computes a two-dimensional position map, we assume that the vertical (z) offsets among participating devices are small. We evaluate the impact of larger z-offsets on Sonoloc's accuracy in Section 7.2.
- All participating devices run the Sonoloc app. Any device running the Sonoloc app can initiate the protocol upon user request, by contacting nearby devices that also run the app via Bluetooth Low Energy.

3.2 Strawman Sonoloc protocol

Consider the following strawman protocol:

1. Select a small number of transmitters randomly from the participating devices.

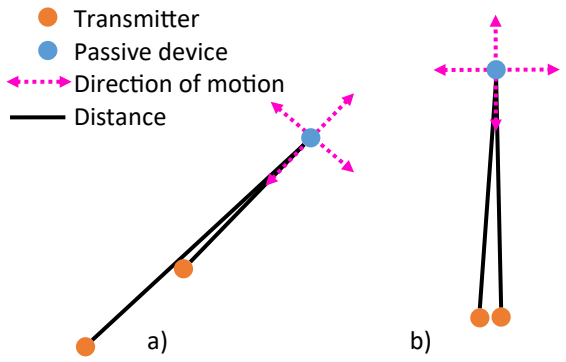

Figure 3.1: Poor transmitter choice examples

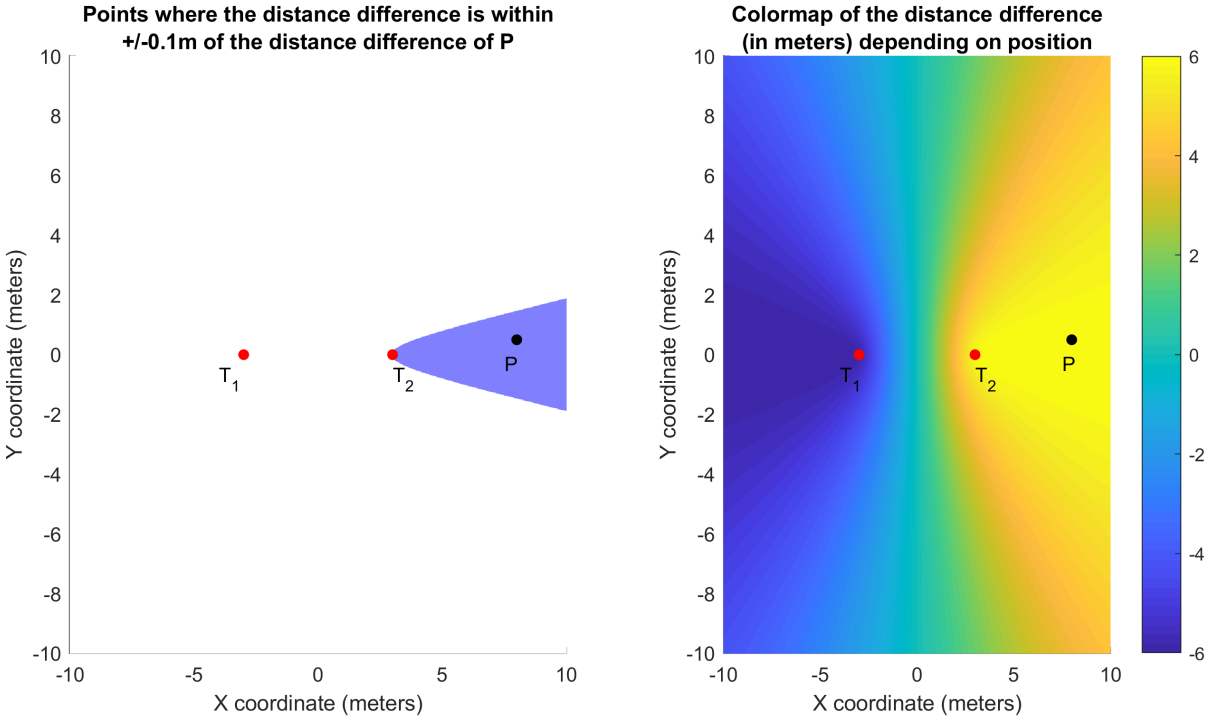

Figure 3.2: Illustration of distance differences for a poor transmitter choice (example a).

2. Emit a chirp from each transmitter in sequence while all devices listen.
3. Determine the chirp arrival times at each device (Chapter 5).
4. Compute the position map from the chirp arrival times measured at all devices (Chapter 6).

This protocol produces position maps with good median accuracy under reasonable audio conditions. However, occasionally, the protocol produces high positioning errors due to a poor choice of transmitters. What is a poor choice of transmitters? Consider the simple examples depicted in Figure 3.1. Recall that a passive device determines its relative position based on the measured *difference* in distance from each pair of transmitters.

Example a): Two transmitters and a passive device lie roughly along a straight line. Although the difference in the distance to the two transmitters is large, if the passive device moved in any

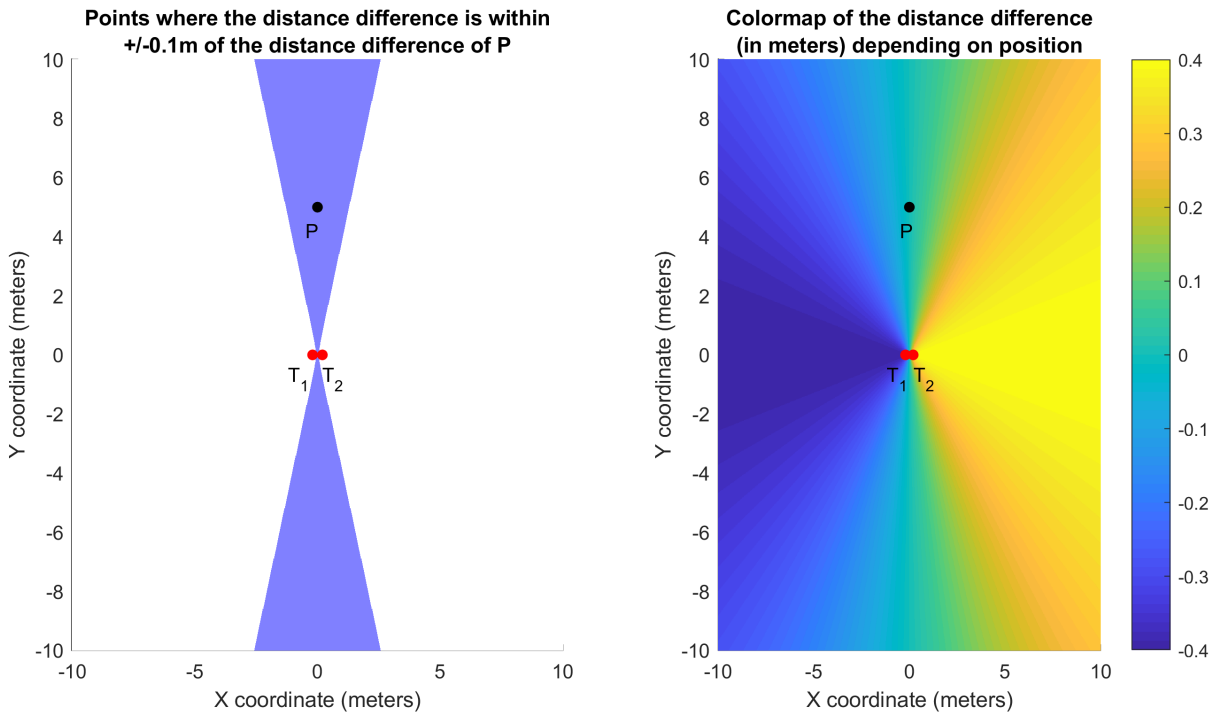

Figure 3.3: Illustration of distance differences for a poor transmitter choice (example b).

direction, it would initially observe very small changes to this difference. Figure 3.2 illustrates this problem. On the left side of the figure, if we move the passive device anywhere within the highlighted area, the distance difference will only change by up to 10 cm compared to the distance difference at the original position. The right side of the figure uses a color map to show the distance difference values, and we can see that a large area around P has the same color. Therefore, accurately positioning the passive device based on this distance difference requires very high-precision measurements.

Example b): Two transmitters are very close, relative to the distance from a passive device. If the passive device moved in any direction, it would initially observe very little change in the distance to the transmitters. Figure 3.3 illustrates the problem in the same way as in the previous example. Note that, on the right side of the figure, the area around P has a very small range of distance differences, and the distance differences in the entire plot are between -0.4 m and 0.4 m, which is much smaller than in the previous example. Again, accurately positioning the passive device requires very high precision measurements.

However, the accuracy of distance and distance difference measurements cannot be improved without bound. In particular, it is limited by the audio sampling rate (at 48 kHz sampling rate, sound travels 7.13 mm during one sample, assuming ambient temperature of 20 °C), line-of-sight obstructions that may lead to the detection of a reflected signal, and z-offsets.

Thus, a *good* transmitter choice would allow for a bit of inaccuracy in the estimated distance differences with only a small effect on *positioning* accuracy. An example of a good transmitter choice is shown on Figure 3.4, which shows that, in contrast to the two poor transmitter choices

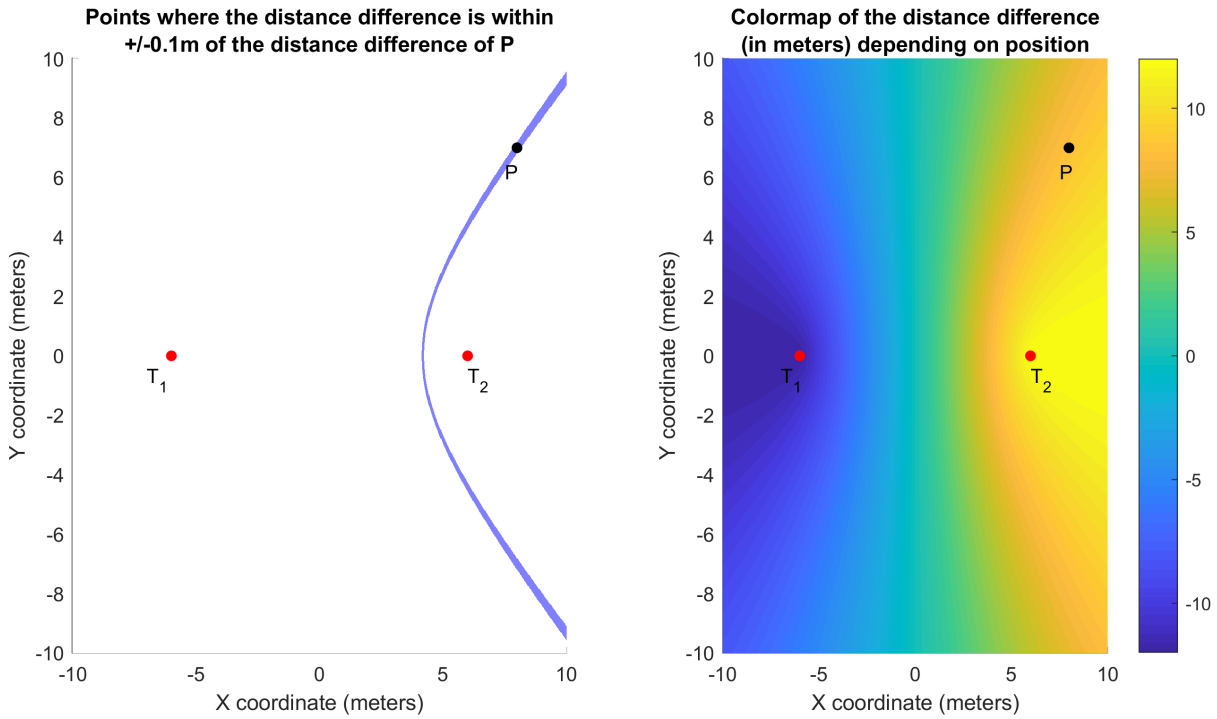

Figure 3.4: Illustration of distance differences for a good transmitter choice.

above, the highlighted area is much smaller. Overall, the transmitters must be well distributed in both dimensions of the space covered by all devices, so that each passive device finds enough pairs of transmitters with large angles between them. The challenge is how to choose a good distribution of transmitters in the absence of a map, which we are trying to produce as the output of the protocol.

3.3 Iterative Sonoloc protocol

We meet this challenge with an iterative transmitter selection protocol. We start with a small set of randomly chosen transmitters and compute a relative position map. This map will not be accurate but provides an approximation sufficient to choose an additional transmitter strategically. In the next iteration, a newly chosen transmitter emits an audio signal, and we re-run the localization to refine the map. (We *do not* re-select the full set of transmitters in each iteration.) We continue this iteration until the desired accuracy is reached.

Choosing the size of the initial transmitter set T_{init} involves a tradeoff between overall protocol delay and the quality of additional transmitter choices. A smaller T_{init} improves our ability to select more transmitters strategically. A larger T_{init} improves the quality of the initial map (which is the basis for the choice of the next transmitter), and reduces overall protocol delay, because T_{init} can be selected in one shot. Empirically, we found $|T_{init}| = 6$ to be a good choice (see Section 7.2.9).

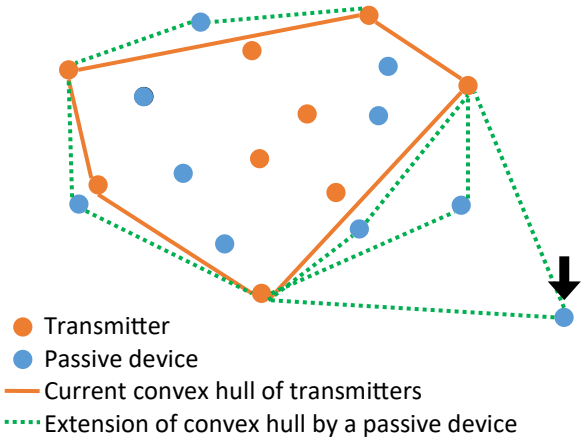

Figure 3.5: Illustration of transmitter selection based on the convex hull. In the example, the bottom right passive device (marked with an arrow) is selected as an additional transmitter.

3.3.1 Choosing additional transmitters

The goal of choosing additional transmitters in each iteration of the protocol is to try to have transmitters along the full range of x and y coordinates occupied by passive devices. Towards this end, we tried a number of different heuristics to choose a set of additional transmitters. The following works well empirically.

In each iteration (other than the first, when transmitters are chosen randomly), we compute the convex hull of the transmitters in the current map. Then, we select a passive device outside the hull that will, if chosen as a transmitter, increase the area covered by the convex hull the most. This step is illustrated in Figure 3.5. Intuitively, by maximizing the area covered by the convex hull of transmitters, we seek to include most passive devices in the hull. A passive device inside the hull is "surrounded" by transmitters, thus avoiding bad transmitter choices like those shown in Figure 3.1. If all passive devices lie inside the convex hull already, we choose additional transmitters from the set of passive devices randomly.

The full Sonoloc protocol is shown in Algorithm 1. We evaluate the number of transmitters (I) required for good accuracy empirically in Section 7.2.

Input: D: set of participating devices
Input: I: desired number of transmitters to use
Output: position map
Select 6 initial transmitters randomly from D;
All devices start recording;
Initial transmitters emit chirps sequentially;
while *#transmitters* < *I* **do**
    Collect measurements, compute map;
    Compute convex hull of transmitters;
    if *all devices in D inside hull* **then**
        Select new transmitter randomly from D;
    else
        Select new transmitter that maximizes area of convex hull;
    end
    New transmitter emits chirp;
end
All devices stop recording;

Algorithm 1: Sonoloc protocol

Chapter 4

Background on signal processing

In this chapter, we provide some background on the signal processing techniques used in Sonoloc. Although these details are not needed to *appreciate* our contributions to the field of localization, they are essential for any *implementation* of a similar system. Sonoloc would not work without these techniques. First, we introduce the concept of signal detection, focusing on the well-known CFAR signal detection algorithm [32] used by Sonoloc. Then, we explain the basics of time delay estimation. Finally, we present a formal definition of Generalized Cross-Correlation with Phase Transform (GCC-PHAT).

These techniques assume that there is only one transmitter and one receiver. Obviously, in Sonoloc, there are multiple transmitters and receivers. However, since we ensure that only one acoustic signal is transmitted at a time, for the purposes of this chapter, it is sufficient to consider a single transmitter and a single receiver.

Before we discuss signal detection, we need to explain what these signals actually look like. Audio signals are continuous, time-varying functions, but digital audio applications represent them in a quantized format, taking samples at fixed time intervals. Accordingly, digital audio data is represented as a list of samples, each of which have an amplitude (e.g., a floating-point number between -1 and 1).

4.1 Single-sample signal detection with CFAR

For a receiver, the detection problem is to decide whether a given audio recording contains a signal. In the simple case where the transmitter sends a very short burst of a single audio sample (called an impulse), it is sufficient to compare the amplitude of the received samples to a simple, pre-defined threshold. Then, if the amplitude exceeds the threshold, the answer is "yes", otherwise it is "no". The next step is to choose the right threshold.

When choosing the threshold, there are two main criteria we want to optimize for. The first one is the detection probability, which refers to the probability with which the detector can correctly recognize the presence of a signal. The second one is the false alarm probability, which

refers to the probability with which the detector incorrectly thinks that a signal is present. In an ideal detector, we want the detection probability to be 1, and the false alarm probability to be 0. However, due to noise in the received data, this is not possible in practice. Choosing a lower threshold increases both the detection probability and the false alarm probability. In contrast, a higher threshold decreases the false alarm probability at the cost of lower detection probability.

A well-known theoretical result for determining the detection threshold is the Neyman-Pearson criterion [61], which allows the user to specify the desired false alarm probability, and use this probability to determine the detection threshold. However, this result assumes white Gaussian noise and known noise power. In practice, however, these two assumptions about the noise do not hold, and the characteristics of the noise may change over time.

A more practical detection method that can handle realistic noise is called CFAR (Constant False Alarm Rate) detection [32].[1] The key idea behind CFAR is that, although the type and power of noise is unknown, we can often assume that, within a certain time window, the noise has relatively constant power, and, beyond a certain time radius of the signal's position, there is no signal. If these assumptions hold, for each received audio sample, called "Cell Under Test" (CUT), the noise power P_{noise} can be estimated using the neighboring cells, which are called training cells. Additionally, to make sure that signal components do not "leak" into the training cells (which could worsen the noise estimate), it is common to use guard cells adjacent to the CUT.

In CFAR, the detection threshold T is given by

$$T = \alpha P_{noise} \tag{4.1}$$

where α is a scaling factor called the threshold factor. It can be shown that, with an appropriate threshold factor α, the resulting false alarm probability can be kept at a constant, hence the name *Constant False Alarm Rate (CFAR)*. In the following, we discuss how to determine the noise power P_{noise} and the threshold factor α.

4.1.1 Estimating the noise power

To estimate the noise power, one possibility is to compute an average based on the training cells. In analog audio, the average noise power is defined as

$$P_{noise} = \frac{1}{T} \int_{a}^{a+T} s^2(t) dt \tag{4.2}$$

where a is the start of the time interval in question and T is its duration. In digital audio, the average noise power can be computed as the mean-square of the amplitudes of a set of audio

[1] The concepts and formulas related to CFAR in this section are based on those found in [32].

samples:

$$P_{noise} = \frac{1}{N} \sum_{i=1}^{N} s(i)^2 \tag{4.3}$$

where N is the number of samples considered, and $s(i)$ is the noise amplitude at position i. The mean-square comes from averaging the individual power, that is, the squared amplitude, of each sample.

4.1.2 Estimating the threshold factor

The other term of the detection threshold is the threshold factor. Given the above noise power estimation method, the threshold factor can be written as

$$\alpha = N(P_{fa}^{-1/N} - 1) \tag{4.4}$$

where N is the number of training cells, and P_{fa} is the desired false alarm rate per received audio sample. The threshold factor can also be expressed in decibels:

$$\alpha_{dB} = 10 \log_{10} \alpha \tag{4.5}$$

4.1.3 Power-to-amplitude conversion

The detection threshold T of Equation (4.1) is the noise power multiplied by a factor, so it is a power threshold for the CUT. To get an *amplitude threshold*, we need to take the square root of the equation, which leads us to a product of (1) the root-mean-square (RMS) of noise amplitudes and (2) an amplitude ratio.

$$T_{amplitude} = \sqrt{T} = \sqrt{\alpha P_{noise}} = \sqrt{P_{noise}} \sqrt{\alpha} \tag{4.6}$$

4.2 Single-sample time delay estimation

Once we have confirmed the presence of a signal in an audio recording, the next step is to estimate, relative to the start of the audio recording, the time delay of the start of the transmitted signal. Differences between pairs of estimated time delays are used to estimate distances and distance differences, which serve as inputs of the localization algorithms.

In an ideal environment with no reverberation, the audio recording contains exactly one signal instance. In this case, this signal is the only one that crosses the detection threshold, and the position of the threshold crossing indicates the time delay of the signal.

In a reverberant environment, sound may be reflected off various objects (ceiling, walls, tables) before it finally arrives at the receiver. The direct path signal will arrive first, followed by reflected signal instances. The delays of the reflected instances are determined by the length of

the path they take, and their strengths are determined by (1) how many times they were reflected, (2) the properties of the objects they were reflected by, and, (3) due to attenuation by air, the total distance they traveled.

Due to reflections, the detector may observe multiple threshold crossing events. Since the time of flight of reflected signals does not match the physical distances, we need to identify the time delay of the direct (shortest) path signal. The corresponding signal instance is the earliest one in the audio recording, but not necessarily the strongest one. As long as the direct path signal is sufficiently prominent, its time delay can be identified. To perform this identification, a number of heuristics have been proposed [20]. Examples include:

- Max: take the position of the maximum value
- P-Max: take the earliest sample among the P highest values
- Simple threshold: take the first threshold crossing event
- Jump Back and Search Forward (JBSF): first, produce a coarse estimation (such as the highest peak), then jump back on the time axis by a predefined amount, and start scanning forward for samples that cross a threshold. This heuristic is based on the assumption that the (potentially weak) peak corresponding to the direct path is located "slightly before" the highest peak.

The "best" heuristic depends on the requirements of the given application.

4.3 Using multi-sample signals

When transmitting a single sample, the power of that one sample limits the total energy of the signal. Since typical mobile audio hardware has only low-power audio transmission capabilities, in practice, they cannot achieve acceptable detection probability and false alarm probability with a single transmitted sample. This limitation can be overcome by transmitting a burst containing *multiple* samples. Since each sample contributes its own energy, the total transmitted energy becomes higher.

However, since the receiver needs to combine information from multiple samples, the receiver's complexity increases. The receiver needs to determine whether there is a (likely distorted) copy of the transmitted pattern (called the "reference signal") among the received audio samples, and estimate the time delay of its first sample.

4.3.1 Energy detection

One way to detect and estimate the delay of a multi-sample signal is known as *energy detection*. This approach uses a narrow-band reference signal. To filter out interference from unused frequencies, the receiver applies a band-pass filter to the recorded signal. Then, to detect the presence of a signal, it looks for an energy burst that exceeds a given threshold. The estimated

time delay is then the position of the first such occurrence within the recorded signal. There are two problems with this approach.

The first problem is that, especially with very narrow-band signals, the bandpass filter may obfuscate the exact start of the signal. Consider an audio recording that contains a finite sine wave signal with a sharp onset (with silence before and after the sine wave). The "sharpness" of the onset cannot be encoded with only the frequency of the sine wave; such an encoding requires all frequencies. The more aggressively we bandpass-filter the audio recording to the sine wave's nominal frequency, the more the onset will lose its sharpness.

The second problem is that, since this approach only looks for the onset and ignores the energy received after the onset, it is not possible to increase the signal-to-noise ratio (SNR) by having a longer signal.

4.3.2 Cross-correlation

The first problem of energy detection can be fixed by using a wideband, flat-spectrum reference signal. To solve the second problem, an alternative approach is needed. A suitable approach is cross-correlation (also known as matched filtering). Cross-correlation is essentially a sliding dot product that slides a known, N-sample pattern (the reference signal) along the (longer) audio recording. At each offset i of the audio recording, it computes the dot product to obtain the correlation coefficient, which represents the "similarity" between the reference signal and the samples $[i, i + N)$ of the audio recording. Then, the signal detector makes its decision by comparing the correlation coefficients to a threshold.

For time delay estimation, the output of the cross-correlation can be used. Without reverberation and without noise, the position of the highest correlation peak indicates the time delay of the reference signal relative to the beginning of the audio recording. In a noisy, reverberant environment, however, the reflected signal instances may be stronger than the direct path signal, so we cannot simply take the highest peak.

Fortunately, as we show in Section 5.2, the reference signal can be designed so that the correlation peak corresponding to each signal instance is sharp enough to resemble an impulse, making the problem similar to what we had in the single-sample case. Then, we can use one of the heuristics for time delay estimation from Section 4.2.

4.4 Generalized cross-correlation

In the following, we formally define Generalized Cross-Correlation with Phase Transform (GCC-PHAT) [37, 13, 9]. Assuming that t represents time, and ω represents frequency, let $x_1(t)$ and $x_2(t)$ be two signals in the time domain and let $X_1(\omega)$ and $X_2(\omega)$ be their Fourier transforms (frequency-domain representations) [33]. Assuming that τ is the displacement, also known as

the lag, the cross-correlation between $x_1(t)$ and $x_2(t)$ is represented as:

$$\phi_{12}(\tau) = \int_{-\infty}^{\infty} x_1(t)x_2(t+\tau)dt, \tag{4.7}$$

which can be calculated efficiently in the frequency domain as:

$$\phi_{12}(\tau) = \mathcal{F}^{-1}[X_1^*(\omega)X_2(\omega)], \tag{4.8}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fourier transform.

The equations above show that cross-correlation can be calculated by multiplication in the frequency domain (by taking the complex conjugate for one signal). If the signals are filtered, an equation similar to (4.8) holds. For example, let $W(\omega)$ be a whitening filter that "flattens" the spectrum to obtain a uniform spectrum. Then, the cross-correlation of the filtered version is obtained by computing:

$$\phi_{12}^{(g)}(\tau) = \mathcal{F}^{-1}[W(\omega)X_1^*(\omega)X_2(\omega)]. \tag{4.9}$$

A commonly used filtering function is the one defined for GCC-PHAT:

$$W(\omega) = \frac{1}{|X_1(\omega)X_2(\omega)|}. \tag{4.10}$$

In this case, $\frac{X_1^*(\omega)X_2(\omega)}{|X_1(\omega)X_2(\omega)|}$ has the unit magnitude for all frequencies and only phase information remains.

The output of GCC-PHAT can be computed in two ways. One is to apply the cross-correlation between the reference signal and the recorded waveform first, and apply the whitening filter afterwards:

$$X_1(f), X_2(f) \xrightarrow{correlation} X_1(f)X_2^*(f) \xrightarrow{whitening} \frac{X_1(f)X_2^*(f)}{|X_1(f)||X_2(f)|} \tag{4.11}$$

The other way is to apply the whitening filter on the input signals of the correlation (reference signal and recorded waveform) first, and then cross-correlate them:

$$X_1(f), X_2(f) \xrightarrow{whitening} \frac{X_1(f)}{|X_1(f)|}, \frac{X_2(f)}{|X_2(f)|} \xrightarrow{correlation} \frac{X_1(f)X_2^*(f)}{|X_1(f)||X_2(f)|} \tag{4.12}$$

The two methods are equivalent. In our implementation, we apply the cross-correlation first, and then we apply the whitening filter on the correlation output.

Chapter 5

Signal processing

The chapter on localization (Chapter 6) shows how to estimate the spatial positions of devices, when we are given signal arrival timestamps as input. However, obtaining sufficiently accurate timestamps is often challenging in the presence of background noise and multi-path interference, especially when transmission and detection use stock smartphone speakers and microphones. In order to address these challenges, we need to solve multiple signal processing problems.

In Sonoloc specifically, we need to address the problems of signal detection, time delay estimation and signal design. The problem of *signal detection* is to determine whether a known audio signal is present in an audio recording or not. However, in order to be able to localize devices, it is not sufficient to confirm the presence of a signal: the localization algorithms require, as input, time difference of arrival between pairs of signals. Thus, the second problem, *time delay estimation*, is to determine these values by precisely identifying the position at which the transmission starts within the audio recording (in other words, the *delay* of the signal relative to the start of the recording). Finally, the third problem, *signal design*, is to design a concrete *reference signal* that, when transmitted and received, enables reliable detection and time delay estimation.

In this chapter, we describe the techniques we use to address these problems. We build on existing techniques, and, in order to address specific issues that arise in Sonoloc, we configure them with Sonoloc-specific parameters, and, in some cases, modify them.

5.1 Signal analysis in Sonoloc

In Sonoloc, signal analysis consists of two stages: (1) computing the correlation of the recorded waveform and the reference signal, and (2) selecting the "correct" peak in the correlation output to detect the start of transmission. Unfortunately, simply choosing the highest peak is not sufficient because high peaks can occur due to reflections and interference. We describe the two stages of signal analysis in turn.

5.1.1 Computing correlation

For computing correlation, we apply a popular variant of Generalized Cross-Correlation (GCC) called Generalized Cross-Correlation with Phase Transform (GCC-PHAT) [37, 13, 9]. Details about GCC-PHAT can be found in Section 4.4. Briefly, after performing the usual cross-correlation, GCC-PHAT executes a frequency-shaping whitening filter phase. The role of the whitening filter is to equalize the overall energy present in each frequency bin by selectively changing the amplitude of each frequency.

This equalization is particularly useful if there are some frequency components that are weaker than the rest of the spectrum, and they contain some weak but clean signal components. In such cases, the equalization makes the signal's time-domain correlation peak more prominent. For example, if, due to the frequency response of the speakers and microphones, some frequency bands are attenuated, the weak frequencies will be enhanced to match the power of the non-attenuated frequencies. As another example, if the noise is loud, but it is concentrated in a subset of the frequency spectrum (typical background noises such as speech and music have this property), the weak frequencies will be enhanced so that their power matches that of the loud noise.

In the context of Sonoloc, two issues arise. The first issue is that, in reverberant environments, the whitening filter phase can cause spurious correlation peaks. In order to mitigate this issue, we need to apply the whitening filter phase of GCC-PHAT via Short-Time Fourier Transform (STFT) [5] instead of standard FFT. The second issue is that the whitening filter does not know which frequencies are used by the reference signal, and it may inadvertently add noise by enhancing unused frequencies. The solution is to band-limit the whitening filter. In the following, we discuss these issues in more detail.

STFT-based whitening filter

During the correlation process, we apply FFT over the entire audio recording. However, because the whitening filter has a very long impulse response (the value at any point of the input can influence the output at any other point), applying the whitening filter over the entire audio recording may lead to spurious peaks in the cross-correlation. More specifically, in cases where (due to reverberation) there are multiple signal instances in the received audio data, there are multiple significant cross-correlation peaks in the unfiltered cross-correlation output. The whitening filter will recognize these peaks as a weak frequency component, and do its best to amplify the weak component. This amplification may lead to spurious peaks in the filtered (whitened) cross-correlation output. (For further discussion about GCC-PHAT's behavior under noise and reverberation, see [90].)

Therefore, we cannot apply the whitening filter to the entire audio recording at once. Instead, we break it up into smaller chunks, and apply the whitening on each of them separately. We apply the whitening filter by using STFT [5]. The STFT is essentially a sliding-window FFT

with overlapping windows. For each window, an FFT is performed, producing a frequency distribution for that window. In our STFT implementation, the number of FFT bins (which affect the resolution of the frequency equalization) is equal to the window length (i.e., the number of samples in the window). In order to reduce artifacts at the window boundaries, the windows typically overlap each other. In order to avoid artifacts when combining the outputs of the overlapping windows, we assign different weights to different positions in the window: we use a Hamming window for analysis and a rectangular window for synthesis. The output of the process is the time-frequency-domain representation of the input signal. This representation is also known as the spectrogram-domain representation.

When applying STFT, we have to choose a window size. By choosing a long window, we can increase the frequency resolution. Conversely, a short window gives better time resolution. This is a kind of "uncertainty principle" that allows us to either have high resolution in time or high resolution in frequency, but not both at the same time. In the following, we explore this tradeoff in the context of Sonoloc.

Since we do not perform time delay estimation in the spectrogram domain, we do not need very high time resolution. Based on the spectrogram data, the whitening filter makes one selective frequency enhancement decision (which frequencies to enhance/weaken) for the entire audio recording (one audio recording contains one signal instance and its reflections), and then applies the same frequency equalization for all STFT windows it is given. Then, it transforms the data back into the time domain.

This single-decision method relies on the assumption that the noise spectrum is approximately constant during the entire recording. Any single frequency will either be amplified or suppressed across the entire recording. Thus, this method is unable to adapt to changes in the noise spectrum within one recording. The non-STFT GCC-PHAT has a similar problem: since it performs a single FFT for the entire recording, it can only make one decision per recording. In the STFT-based GCC-PHAT, the spectrogram-domain representation given by the STFT allows up to one decision per STFT window. However, finding the optimal granularity of frequency enhancement decisions requires further evaluation and it is beyond the scope of this thesis.

An upper bound for the window length can be derived from the following. If the window is long enough to include the correlation peaks of multiple signal instances, then the peaks within the window will produce an interference pattern in the spectrogram, which confuses the frequency enhancement algorithm. This effect is most observable if there are multiple, regularly spaced reflections. As a result, spurious correlation peaks will appear and/or existing peaks will be suppressed, making time delay estimation difficult. In order to prevent this from happening, we should ensure that the STFT window is short enough not to include the peaks of multiple signal instances, i.e., it should be shorter than the expected distance (in audio samples) between two signal instances in a recording. In situations where signals from phones held just a bit above a table can be reflected off of said table, there can be reflections around 1 m longer than the shortest path. At the nominal speed of sound at 48 kHz sampling rate, this corresponds to

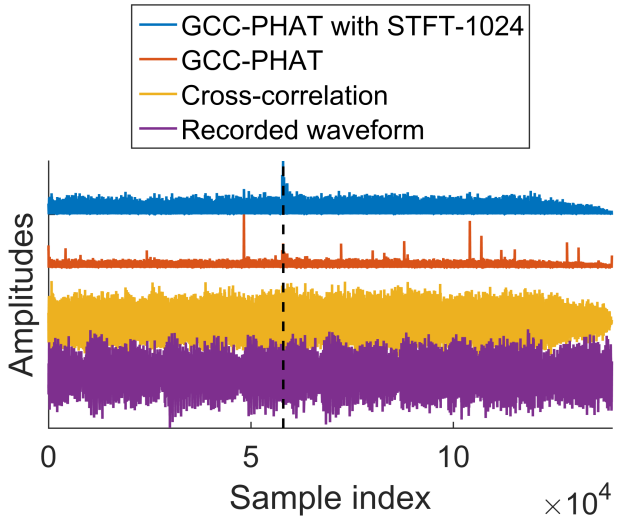

Figure 5.1: Audio recording of signal and background music, and different correlation signals. The signal starts at the dashed vertical line.

around 140 audio samples or 2.19 ms. Therefore, the window should be shorter than this value.

In Sonoloc, we chose an STFT window length of 128 samples, which supports 128 frequency bins, i.e., given a 24 kHz spectrum, frequency enhancement decisions are made at a resolution of 187.5 Hz.

To illustrate the advantages of GCC-PHAT+STFT over regular GCC-PHAT and regular cross-correlation, Figure 5.1 shows an example comparing GCC-PHAT+STFT to these other methods. The audio waveform is a phone recording of a signal emitted by another phone in the presence of loud background music. In this example, regular cross-correlation (or GCC-PHAT with full FFT) does not produce a clear spike in the time domain, but GCC-PHAT with STFT does.

Band-limited whitening

By default, the whitening filter considers the whole spectrum for frequency enhancement. However, it is not a good idea to enhance frequencies that, by design, do not contain any signal component. A band-limited reference signal obviously leaves a portion of the spectrum unused. Since these unused frequencies can only contain noise, amplifying them would only make things worse.

To get around this problem, we need a band-limited whitening process that focuses on only the frequencies that are actually present in the reference signal. We implement this process as follows:

1. Completely filter out the unused frequencies by setting their amplitudes to 0 in the frequency domain.
2. Apply the whitening filter on the remaining frequencies by measuring and equalizing their

relative energy.

5.1.2 Signal detection and time delay estimation

After the correlation stage, the next two tasks are *detection* (did we receive a signal?) and *time delay estimation* (if we did, where does it start?). First, we focus on signal detection. Taking the output of the cross-correlation as input, Sonoloc performs signal detection based on a technique called CFAR (Constant False Alarm Rate).

We discuss CFAR in more detail in Section 4.1. Briefly, CFAR takes a list of data samples and inspects each of them as the Cell Under Test (CUT), searching for any sample that stands out from the set of neighboring (noise) samples with its high amplitude. To determine whether a sample stands out, CFAR compares it to a detection threshold. The detection threshold is the product of (1) an estimation of the noise power based on a subset of the neighboring samples (so-called *training cells*) and (2) the *threshold factor*. The threshold factor is a function of the number of training cells and the false alarm rate.

Choosing an appropriate threshold is critical. If the threshold is too low, a false alarm may occur potentially hundreds of milliseconds before the direct path signal, which renders the corresponding distance estimates useless. If the threshold is too high, then CFAR will miss the direct path signal, and, depending on the strength of reflected signals, either the detection will fail, or we will mistake a strong reflected signal for the direct path signal. (Strong reflected signals could appear, for instance, if the direct path signal was weakened by an obstacle but some reflected signals propagated freely, and they can be further strengthened by constructive interference.) Since, in rooms with strong reverberation, reflected signals can appear tens of milliseconds after the direct path signal, such a mistake would render the corresponding distance estimates useless.

The detection threshold can be controlled via CFAR's configuration parameters, which include the layout of the training cells and the desired false alarm rate (the latter of which is a distinguishing feature of CFAR).) Below, we focus on choosing the values for these parameters in the context of Sonoloc.

Training cell layout

The number of training cells is subject to a tradeoff; while a small amount of training cells allows the system to adapt to changes in the background noise level quickly, a large amount of training cells improves the quality of our noise power estimate. Empirically, we found that around 1000 training cells work well. (At 48 kHz sampling rate, this corresponds to about 21 ms.)

Next, we decide on the *position* of the training cells relative to the CUT. Sonoloc only needs data from the direct path signal, i.e., the one that travels the exact distance between transmitter and receiver. When the CUT is the sample corresponding to the start of the direct path signal, the audio samples before the CUT should contain only noise, while those after the CUT may

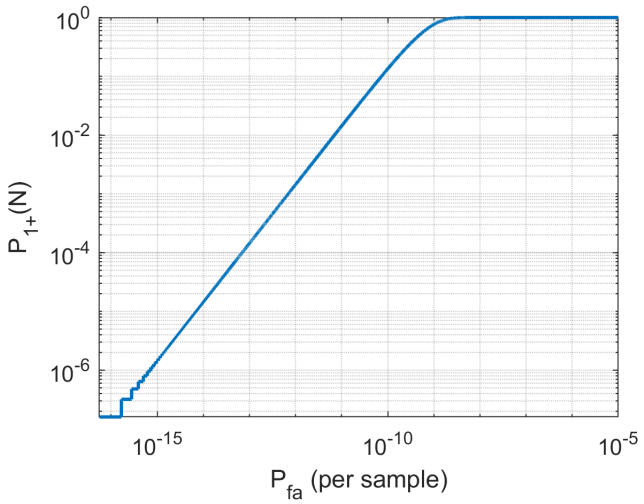

Figure 5.2: Probability of at least one false alarm per 1.44×10^9 samples as a function of P_{fa}.

be contaminated due to sound being bounced off objects and walls in the room. Since such contamination can negatively affect the noise power estimate, Sonoloc only uses *past* samples as training cells. Additionally, since correlation peaks are not always single-sample peaks (especially if the "true" start of the signal falls in between two samples), 2 cells right before the cell under test (CUT) are designated as guard cells. In summary, the cells in the interval [CUT-1000, CUT-2) are training cells, and CUT-2, CUT-1 are guard cells.

False alarm probability

In CFAR, the false alarm probability (let us call it P_{fa}) refers to the probability that a false alarm will occur on *a single audio sample*. However, to create a position map, a large number of samples are needed, and even a single false alarm among them can affect the localization accuracy. A conservative estimate is 15 audio transmissions, 500 devices, 2 microphones each, 2 seconds per recording and a sampling rate of 48 kHz, which would result in 1.44×10^9 total recorded samples. Thus, we need to make sure that, with high probability (close to 1), none of those audio samples produce a false alarm. Let $P_{1+}(N)$ be the probability of that, during the inspection of N samples, at least 1 false alarm will occur. This probability can be computed as:

$$P_{1+}(N) = 1 - (1 - P_{fa})^N \tag{5.1}$$

We should make sure that this probability is close to 0. Figure 5.2 shows the relationship between P_{fa} and $P_{1+}(N)$ on a log-log plot. Based on the figure, $P_{fa} = 10^{-15}$ gives $P_{1+}(N) = 1.4388 \times 10^{-6}$, which we consider to be good enough.

Putting it together

In Sonoloc, we perform a CFAR-based signal detection on the output of the cross-correlation. For the above training cell layout and false alarm probability, CFAR produces a threshold factor of around 15.46 dB. We found this to be a little too high, thus, in our implementation, we use a threshold factor of 15 dB. We call this threshold the *local threshold*.

We also set a *global threshold*, again 15 dB above the estimated noise level. The global threshold uses all samples in the audio recording as training cells, which helps ensure that the detected correlation peak is significant compared to a global noise estimate. Since the global noise power estimation is based on the entire audio recording, it also includes the peaks corresponding to both direct and reflected signal instances. However, the impact of contamination caused by reflections is negligible due to the much larger number of samples than in the case of the local threshold.

Additionally, sometimes, correlation peaks come in small bursts with increasing and then decreasing amplitude. (We suspect that this is a side effect of the whitening filter in GCC-PHAT under noisy conditions.) Sonoloc tries to get to the top of the burst by requiring that the peak is a local maximum within a 60-sample radius of the peak. This radius needs to be larger than burst region's radius (in our experience, up to about 50 samples), and it has to be smaller than the minimum observed distance in samples from direct path to 1st reflection (in our experience, 65 samples or more).

Our signal detection condition is the conjunction of all 3 conditions (local threshold, global threshold, local maximum). It is possible that the correlation output does not contain such a prominent peak; in this case, the detector signals a failure.

The final step in the signal analysis process is to estimate the time delay. We are interested only in the time delay of the direct path signal. Assuming that the direct path signal is strong enough, we want to estimate its time delay correctly, regardless of how strong the subsequent reflections are. From many possible heuristics that have been proposed (some of them are mentioned in Section 4.2), the one that matches the idea of "looking for the first thing that looks like a signal" is to take the first threshold crossing event, so we chose this approach in Sonoloc. More precisely, given a correlation output, we estimate the time delay as the index of the first sample that satisfies all detection conditions.

5.2 Signal design

Since Sonoloc relies on a form of cross-correlation to perform signal analysis, in order to be able to estimate the time delay of the first sample of the reference signal unambiguously, the reference signal has to produce a single, strong, sharp peak in the correlation output. The quality of the correlation peak strongly depends on the properties of the reference signal. In this section, we discuss the properties of an ideal reference signal, and finally decide on a concrete signal.

5.2.1 High signal-to-noise ratio (SNR)

For good *signal detection* performance, the signal waveform should be chosen so that it maximizes the signal-to-noise ratio (SNR) at the receiver, i.e., the correlation peak at the receiver should stand out from the noise as much as possible. The SNR can be increased by transmitting more energy, which can be done by transmitting at the same power for a longer time, or by transmitting at a higher power within the same amount of time. It is feasible to use both techniques simultaneously.

Signal length: Assuming constant signal power, increasing the signal length increases the total transmitted energy, increasing the SNR at the receiver. However, for the cross-correlation to work correctly, the devices must not move during transmission, which, in our case, effectively limits the signal length to a few seconds. Additionally, if the signal uses audible frequencies, increasing the signal length excessively would make the protocol obtrusive and annoying for users.

Signal power: The reference signal's power is essentially a sum of squared amplitudes for all signal samples, thus it can be increased by using a signal with a low crest factor. The crest factor of a signal is defined as the ratio of the peak of the signal to the root-mean-square (RMS) amplitude. The peak amplitude of the signal waveform is limited by the digital audio format. At the same peak amplitude, a signal with a low crest factor has a higher power than a signal with a high crest factor, so it can produce higher SNR [14].

The SNR is also influenced by other factors that are beyond the system designer's control. These factors include distance (due to attenuation), background noise, the characteristics of the audio hardware (e.g., frequency response, microphone sensitivity), and any objects that at least partially obstruct the line of sight between transmitter and receiver.

5.2.2 Single, sharp correlation peak

Following standard practice, we evaluate reference signal candidates by examining their autocorrelation output. The autocorrelation output is obtained by cross-correlating the reference signal with itself. The output of autocorrelation tells us what the cross-correlation would look like under noise-free conditions. If there is no noise, no reverberation, and devices have flat frequency characteristic for sound emission and reception, the recorded waveform would be a time-delayed and amplitude-decayed version of the transmitted (reference) signal, and the cross-correlation between the recorded waveform and the transmitted signal would be identical in shape to the autocorrelation of the transmitted signal.

In order to be able to estimate the *time delay* unambiguously, the signal's autocorrelation must produce exactly one significant peak. Therefore, the reference signal must not be periodic, i.e., there should be no similarity between parts of the signal. Additionally, in order to avoid

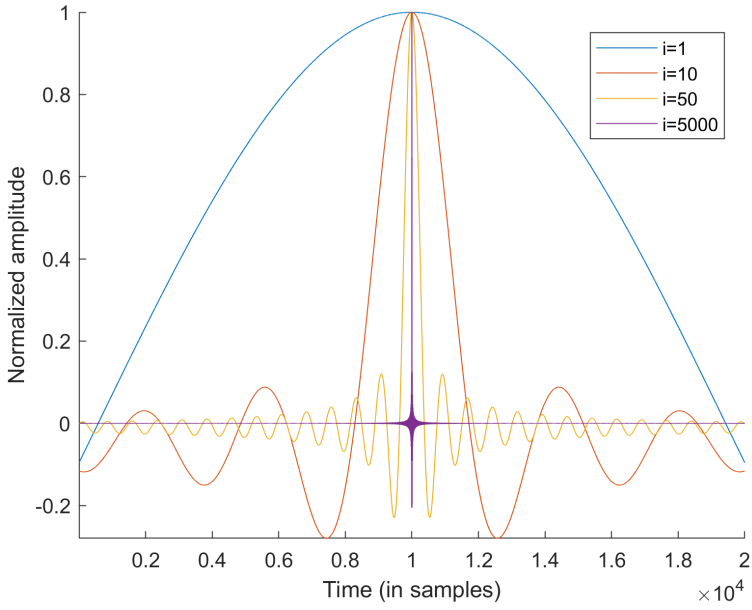

Figure 5.3: Illustration of approximating a sharp peak with an increasing number of frequency components.

significant peaks due to background noise, the reference signal should not be similar to typical background noise (e.g., human speech, music).

Even if the autocorrelation has only one significant peak, in real environments such as reverberant rooms, additional peaks will be generated due to receiving signal reflections. These peaks may follow the direct path signal very closely. When using a signal that has a "gentle" autocorrelation peak, such as a single sinusoid, the peaks may merge, limiting the resolution at which the peak can be detected. Therefore, the correlation peak must be sharp.

In order to have a sharp correlation peak, the reference signal must be wideband. Figure 5.3 shows an iterative approximation of a sharp peak. We start from a single cosine wave ($i = 1$), and, in each iteration, we add another cosine wave with a higher frequency. We can see that, the higher the number of frequency components (i), the sharper the peak becomes. Thus, the correlation output needs to be wideband. Since, as we will illustrate below, the correlation process retains the signal's spectrum, the reference signal should also be wideband.

In addition to being wideband, the reference signal should have a nearly flat spectrum. There are two reasons for this. First, just like missing frequency components, very weak frequency components limit the maximum achievable sharpness of the peak. Second, the whitening filter of GCC-PHAT will equalize the unevenness of the spectrum anyway.

To illustrate the advantages of wideband signals, Figures 5.4, 5.5 and 5.6 show visualizations of a linear frequency sweep, a simple sine wave and a wideband pseudorandom signal. By comparing the "signal FFT" and "autocorrelation FFT" subplots, which show the power spectral density of the signals, we can see that the signal's frequency band remains intact after the correlation process. The figures also show that, unlike the sine wave's autocorrelation (which includes a strong, high-frequency oscillation not discernible on the figure), the two wideband

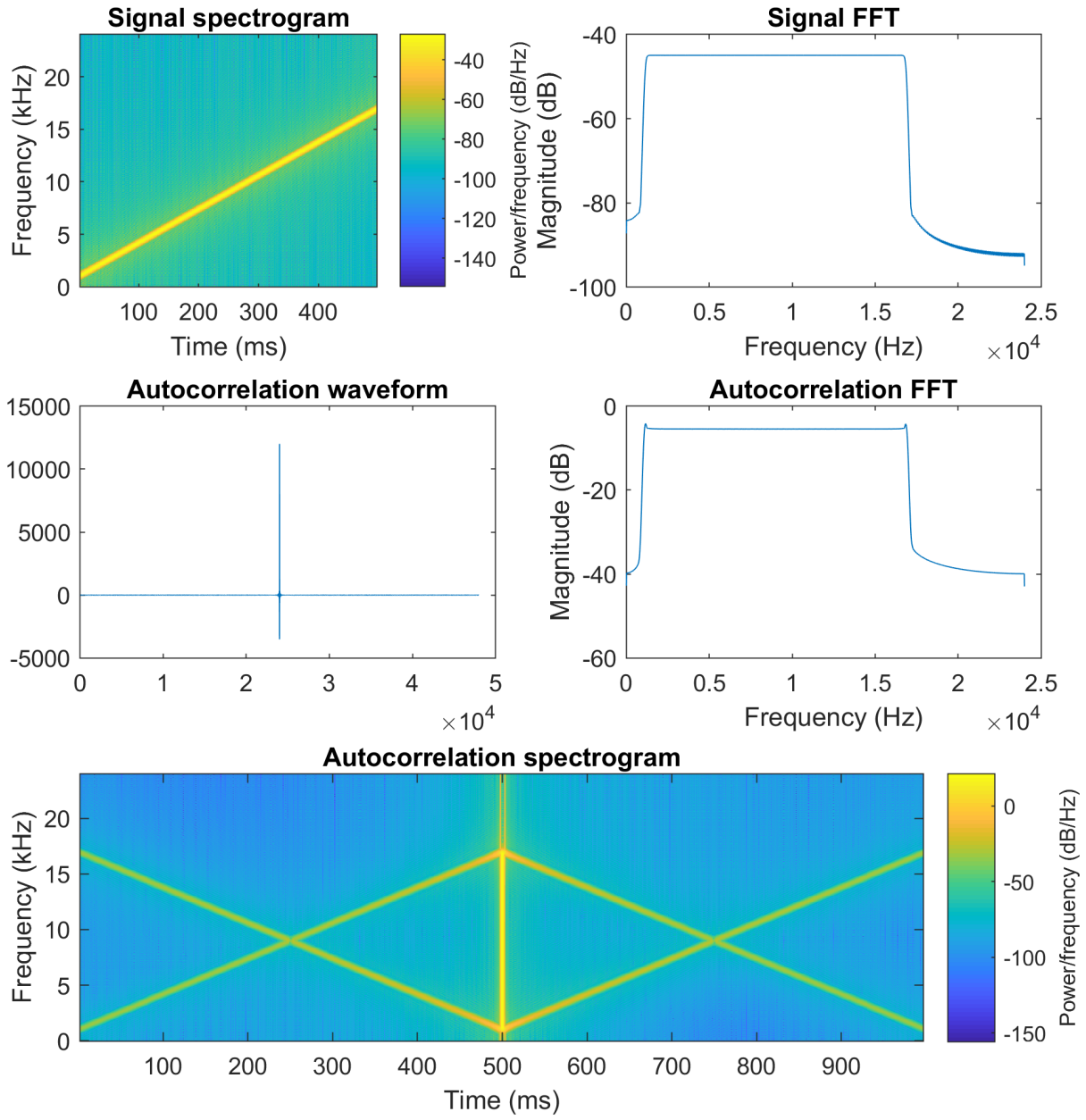

Figure 5.4: Visualization of a linear frequency sweep and its autocorrelation.

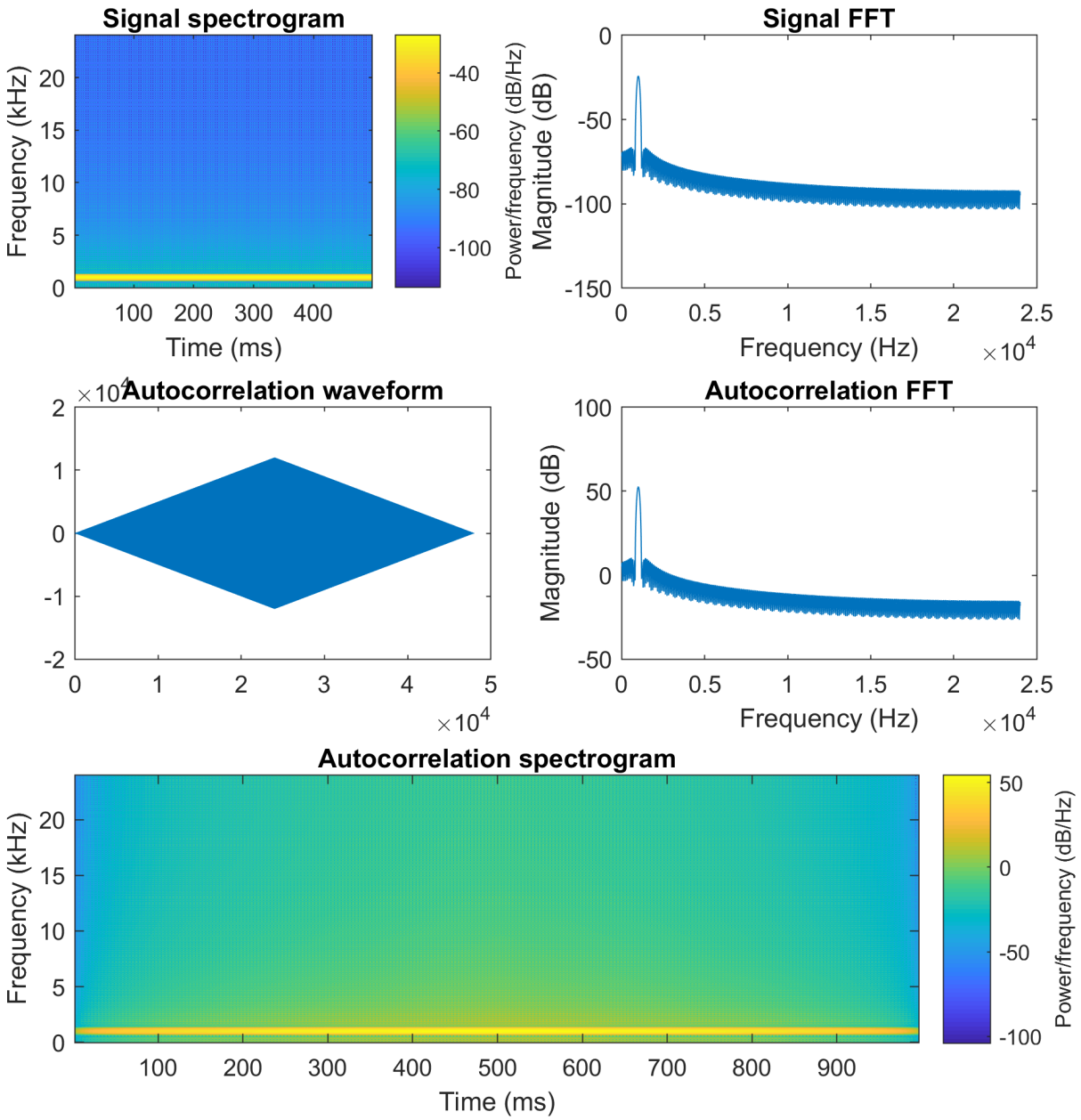

Figure 5.5: Visualization of a sine wave and its autocorrelation.

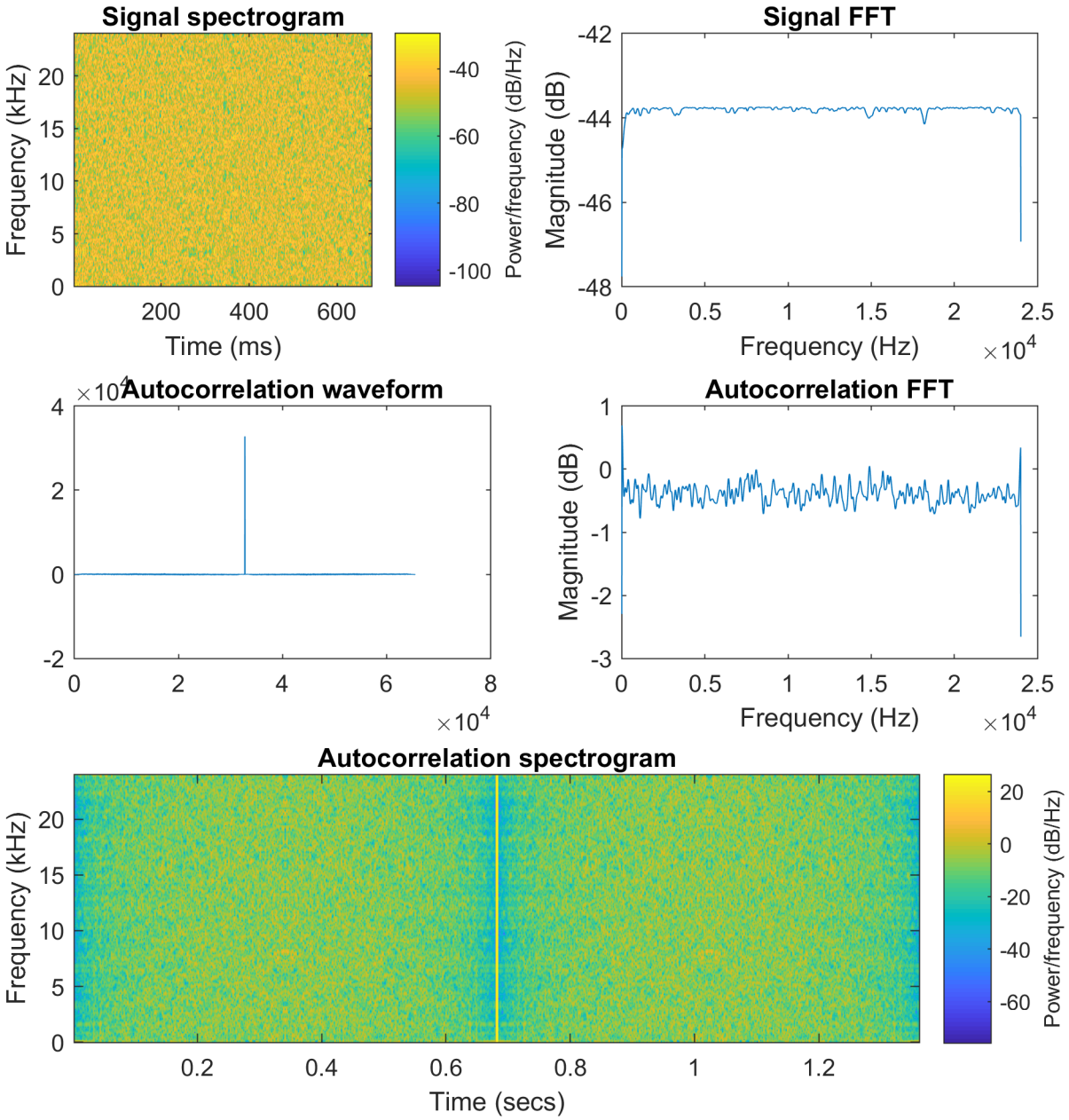

Figure 5.6: Visualization of a wideband pseudorandom signal (M-sequence) and its autocorrelation.

signals have a single, sharp autocorrelation peak.

Finally, the reference signal should not include frequencies on which the audio hardware's frequency response is very weak (such as the 0 kHz–1 kHz and 17 kHz–24 kHz frequency bands on our phones). At the receiver, such frequency bands will be much weaker than the rest of the spectrum and their SNR will be very low. The whitening filter of GCC-PHAT will amplify these weak frequency components, but, due to their very low SNR, the amplification might decrease the overall SNR. While it is possible to adjust the whitening filter so that it does not amplify the frequency bands that are known to have this problem, it is better not to include these frequencies in the reference signal in the first place.

5.2.3 Choosing a signal

Among the signals that satisfy all of our requirements, there are two commonly used signal types. One is maximum length sequences (M-sequences), and the other one is chirp signals.

M-sequences [25, 92, 24, 26] are wideband pseudorandom sequences that have very desirable autocorrelation properties. The autocorrelation of an M-sequence is 1 for zero lag (time delay), and nearly zero ($-1/N$ where N is the sequence length) for all other lags; in other words, the autocorrelation of the M-sequence approaches the unit impulse function as the sequence length increases. Thus, M-sequences are theoretically optimal, but they contain all frequencies, including frequencies that are difficult to reproduce with sufficient power on commodity smart devices. In order to restrict the sequence to the desired spectrum, we can apply a bandpass filter.

An alternative to M-sequences is a chirp signal [17]. In radar and sonar applications, the most typically used signals to achieve pulse compression [18, 49] are linear chirps. A linear chirp is a band-limited signal that contains all frequency components from the specified starting frequency to the specified ending frequency with the same amplitude. It forms a time-domain sinusoidal waveform with linearly changing frequency, ensuring that the signal is not periodic.

Both are valid choices; we decided to use a chirp signal.

5.2.4 Chirp configuration

For a chirp, the main parameters to configure are the signal length and the frequency range. Regarding the signal length, our experiments suggest that a 500 ms long signal is sufficient to provide adequate SNR in practice. As for the frequency range, we have experimentally observed that the lower and upper end of the supported acoustic spectrum, specifically the 0 kHz–1 kHz and 17 kHz–24 kHz frequency bands, are much weaker than the middle frequency band. (Due to the small speaker size, low frequencies are expected to be weak.) Thus, we chose the chirp's frequency boundaries to be 1 kHz and 17 kHz. In summary, our reference signal is a linear upward frequency sweep (chirp) from 1 kHz to 17 kHz with a length of 500 ms. Figure 5.7 shows its spectrogram, and Figure 5.8 shows its autocorrelation.

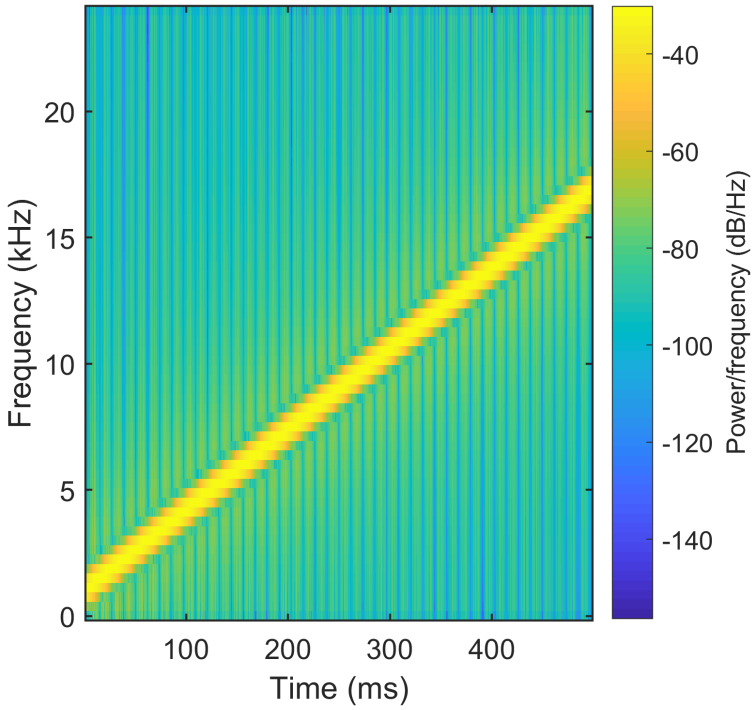

Figure 5.7: Reference signal spectrogram

This frequency band is audible, and, in some scenarios, users might find the chirps annoying. A potential solution would be to use ultrasound. However, commodity smartphones have difficulty transmitting and receiving sounds above 17 kHz, let alone higher-frequency ultrasound. Frequencies above 24 kHz would also require a sampling rate higher than 48 kHz, which is not available on commodity smartphones either. That said, we could regain the SNR we lost due to the lower sensitivity by transmitting a longer signal in the 17 kHz–24 kHz frequency band. In our experience, matching our standard reference signal's SNR would require a high-frequency signal of 3 or more seconds. However, there are several problems with longer signals. First, the total duration of the protocol increases significantly, and users' devices must remain stationary during that period. Second, due to the unstable audio sampling rates of smartphones, the amount of clock drift that can accumulate during the protocol also increases, negatively affecting the accuracy. Additionally, high-frequency sounds are attenuated more easily than low-frequency sounds. For these reasons, we leave the investigation into inaudible reference signals as future work.

So far, we have discussed what the reference signal itself should look like. Next, we discuss how it should be transmitted.

5.2.5 How to transmit the signal

The main choice when transmitting a fixed-waveform signal is the transmission volume. Since a louder signal provides higher SNR, it may be tempting to transmit at the highest possible

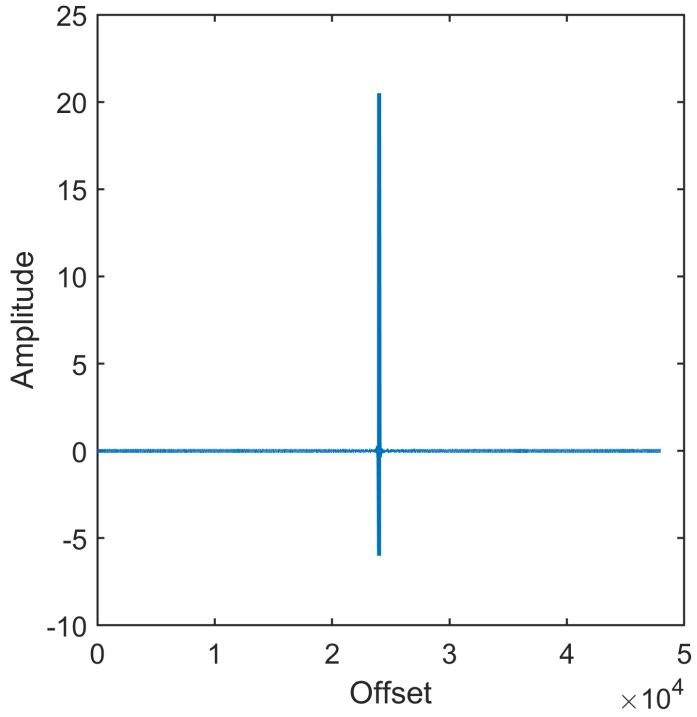

Figure 5.8: Reference signal autocorrelation

volume. However, we need to watch out (1) for signal distortions caused by the audio hardware and (2) for potential damage to the human ear.

Distortion can happen during signal transmission or during signal recording. During signal transmission, distortion occurs when the amplifier amplifies the signal beyond the capabilities of the speaker. In devices with fixed audio hardware (such as smartphones with built-in speakers), the digital playback process can be calibrated in a way that, even at the highest amplitude allowed by the digital audio format, it only produces signals within the design specification of the speaker. Based on experiments with our phones, we concluded that the speaker's distortion is negligible compared to the additional SNR provided by the increased transmission power.

Distortion can also happen at the receiver. When the amplitude observed at the receiver is so high that it exceeds the range defined by the receiver's digital audio format, the recorded audio waveform will be distorted, decreasing the SNR at the receiver. This phenomenon is called clipping. In our experiments, even at the highest possible transmission volume, clipping only occurred in cases where the microphone and speaker belonged to the same device. Since these signals travel less than 20 centimeters, the SNR observed at the microphone is so high that, despite the decrease caused by clipping, time delay estimation is still possible. Therefore, we conclude that, in practice, it is okay to transmit at the highest volume.

A third form of distortion happens if a speaker is not sufficiently "warmed up". For instance, it may be in a power-saving mode when it receives the transmission command, and it may take a short time to become fully operational. In the meantime, it may produce distorted sound (or

no sound at all). One of our studio speakers exhibited this issue. In order to warm up such a speaker, we need to send some arbitrary audio data to it for a short time (up to a few hundred milliseconds). On our phones, we did not observe any difference in the quality of the correlation output regardless of whether a warmup sequence was used.

Finally, we consider the potential damage to the human ear. On our devices, a chirp transmitted at the maximum transmission volume produces an A-weighted peak sound pressure level of 93 dB at the speaker, and 78 dB 50 cm away from the speaker. According to the Occupational Safety and Health Administration, a sound pressure level of 95 dB is permissible for 4 hours per day [74]. This limit on exposure time is orders of magnitude higher than the time required for Sonoloc's audio transmissions.

Chapter 6

Localization

This section describes how Sonoloc computes a position map from the measured times of arrival determined by the signal detector from Chapter 5. We begin with a description of how Sonoloc determines distances and distance differences from the signal arrival times. Then, we describe the algorithm for determining transmitter locations from their pairwise differences. We close with the algorithm for determining the positions of passive devices from their measured distance differences to the transmitters. We use the notation in Table 6.1. The relevant distances are illustrated in Figure 6.1.

6.1 Estimating distances and distance differences

Using the acoustic signal processing techniques described in Chapter 5, each device determines the time at which an audio signal (including its own transmitted signal) captured by its microphone starts. At first, the arrival time of a signal is expressed in terms of the receiver's local audio sampling clock, i.e., it is an index in a list of received audio samples. To express it in *seconds*, as required by the equations below, the sample index must be divided by the audio sampling rate in Hz.

Sonoloc uses these independent arrival timestamps to derive two types of possible observations: (i) distances between the speaker in one transmitter and the microphone in another transmitter, and (ii) distance differences given by a triple of two speakers in two transmitters and a microphone in a passive device. The protocol only uses arrival time differences of signals captured by the same device, therefore the audio sample clocks need not be synchronized.

Symbol	Meaning
d_{xy}	Distance from x's speaker to y's microphone
t_x	Global time at which x transmitted a signal
r_{xy}	Global time at which x's signal was received at y
f	Sampling rate (Hz)
c_s	Speed of sound (343.2 m/s @ 20 °C)

Table 6.1: Notation; times are expressed in terms of a (hypothetical) global clock.

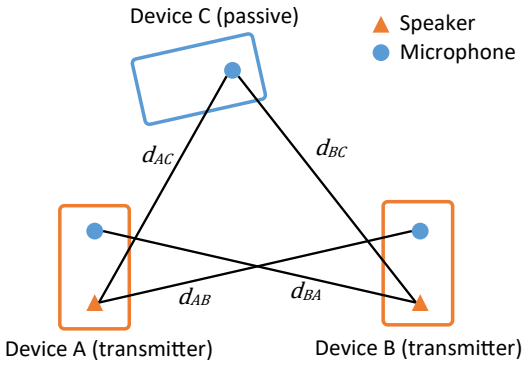

Figure 6.1: Distance notation

6.1.1 Estimating distances

Estimating the distance between a pair of devices is not as simple as measuring time of flight of an acoustic signal directly because (1) we cannot assume synchronized clocks and (2) there are unknown delays in the audio software stack, which makes it difficult to obtain accurate transmission and reception timestamps. To overcome this problem, Peng et al. proposed an algorithm called BeepBeep [65]. In BeepBeep, both devices transmit an acoustic signal and both devices measure the difference between times of arrival of the 2 signals. Then the devices exchange these differences wirelessly. Based on these differences, BeepBeep estimates a single time of flight, and, by multiplying it by the speed of sound, it estimates the distance between the two devices.

We generalize the BeepBeep algorithm to estimate, for all pairs of transmitters, the distance between the speaker in one transmitter and the microphone in another transmitter. In contrast to repeating the BeepBeep technique for all transmitter pairs (which would require n^2 audio transmissions for n transmitters), the below generalization requires only one audio transmission per transmitter, leading to a total of n transmissions for n transmitters. Assume that a set of transmitters have emitted an acoustic signal. For notational convenience, we assume a hypothetical global clock for the time equations below, but none of the computations at any device require a global clock. Instead, like in BeepBeep, each device only computes *intervals* based on their local recorded timestamps.

Given a signal emitted by a speaker in a device A at time t_A according to a global clock, the following two equations hold for its reception at devices A and B, respectively:

$$r_{AA} = t_A + \frac{d_{AA}}{c_s} \tag{6.1}$$

$$r_{AB} = t_A + \frac{d_{AB}}{c_s}. \tag{6.2}$$

Note that, for convenience, we are defining the time at which A's signal is received at B in terms of when A sends its message. Devices can be localized without knowing the value of t_A.

($t_A - t_B$, used later, can be determined even if t_A and t_B are unknown.)

Similarly, for a signal emitted by a speaker in a device B, the following holds:

$$r_{BB} = t_B + \frac{d_{BB}}{c_s} \tag{6.3}$$

$$r_{BA} = t_B + \frac{d_{BA}}{c_s}. \tag{6.4}$$

Given equations (6.1), (6.2), (6.3), and (6.4), we eliminate t_A and t_B so that we obtain:

$$d_{AB} + d_{BA} = d_{AA} + d_{BB} + c_s(r_{BA} - r_{AA} + r_{AB} - r_{BB}) \tag{6.5}$$

If d_{AA} and d_{BB} are small compared to d_{AB} and d_{BA}, we can use $(d_{AB}+d_{BA})/2$ to estimate both d_{AB} and d_{BA}, the distance between A's speaker and B's microphone, and the distance between B's speaker and A's microphone, respectively.

If a signal fails to be detected on all microphones of a transmitter, the associated pairwise distance will be missing. In this case, we can estimate the missing value as follows: for all triples of transmitters involving the pair with the missing distance, compute bounds based on the triangle inequality. However, in our experiments, signal detection failures did not occur on all microphones of a device simultaneously.

6.1.2 Estimating distance differences

Suppose that A and B are transmitters (as above) and C is a passive device that locally timestamps the audio signals from A and B. Our goal is to derive the distance difference $\tau_{AB|C} = d_{AC} - d_{BC}$. Similar to equations (6.1), (6.2), (6.3), and (6.4), it is clear that

$$\begin{aligned} r_{AC} &= t_A + \frac{d_{AC}}{c_s} \\ r_{BC} &= t_B + \frac{d_{BC}}{c_s}. \end{aligned} \tag{6.6}$$

Thus,

$$\tau_{AB|C} = d_{AC} - d_{BC} = c_s(r_{AC} - r_{BC} - (t_A - t_B)) \tag{6.7}$$

Device C can directly measure the time difference $r_{AC} - r_{BC}$ locally. The quantity $t_A - t_B$ can be estimated at device B as follows:

$$t_A - t_B = \frac{d_{BB} - d_{AB} + c_s(r_{AB} - r_{BB})}{c_s} \tag{6.8}$$

Note that the equation requires d_{AB}, which we can estimate using $\frac{(d_{AB}+d_{BA})}{2}$ as long as the distance between devices is much larger than the distance between speaker and microphone on the same device.

In general, $\tau_{AB|C}$ can be estimated only when A and B are "synchronized" to each other in the sense that C knows the offset between the two transmissions in *global* time. The offset allows C to adjust its reception timestamps to be as if the transmissions of A and B happened at the exact same time, which makes C's observed TDOA values proportional to the distance difference. Instead of assuming synchronization, Sonoloc utilizes *transceiver* nodes (A and B) equipped with a speaker and a microphone at almost the same position to estimate the offset $t_A - t_B$ and perform the synchronization implicitly.

Finally, in order to reduce the impact of noisy distance differences (due to inaccurate signal detection) on the localization accuracy, Sonoloc filters the estimated distance differences based on the triangle inequality. According to the triangle inequality, $\forall A, B \in Transmitters : \forall C \in Passive : |\tau_{AB|C}| < d_{AB}$. We compute the set of device triples that violate the triangle inequality by a given tolerance factor F:

$$\{(A, B, C) \mid A, B \in Transmitters \wedge C \in Passive \wedge |\tau_{AB|C}| \geq F \cdot d_{AB}\} \tag{6.9}$$

For each such triple, we ignore the associated distance difference constraint in the subsequent localization.

In sensitivity experiments, we found that this filtering is required for good localization accuracy. The value of F needs to be low enough to filter out high-error distance difference measurements that negatively affect the results. Empirically, we determined that $F = 1.2$ is a good choice (see Section 7.2.9).

6.2 Computing coordinates

Based on the estimations described above, the localization phase produces two-dimensional (2D) coordinates for all participating devices in Sonoloc. We use two types of localization:

1. Transmitter localization via joint speaker-microphone localization based on the distances estimated in Section 6.1.1
2. Passive device localization via microphone localization based on (1) the distance differences estimated in Section 6.1.2 and (2) the coordinates of the transmitters computed in step 1.

More precisely, for a device A, let $\mathbf{s}_A$ be the position of A's speaker, and let $\mathbf{m}_A$ be the position of A's microphone. Since our goal is to localize *devices* (and, indirectly, their owners), we are interested neither in determining the orientation of the devices nor in localizing speakers and microphones individually. Thus, we define the position of a device A as follows. If A is a transmitter, we define its position as the average of the coordinates of its speaker and microphone, i.e.,

$$pos_A = \frac{1}{2}(\mathbf{s}_A + \mathbf{m}_A). \tag{6.10}$$

If A is a passive device, we define its position as the coordinates of its microphone, $\mathbf{m}_C$. The first localization estimates $\mathbf{s}_A, \mathbf{m}_A$ for all transmitters A based on *estimated* distances given for all pairs of transmitters A, B based on Section 6.1.1:

$$\hat{d}_{AB} \simeq d_{AB} = \|\mathbf{s}_A - \mathbf{m}_B\| \tag{6.11}$$

where $\|\cdot\|$ is the Euclidean distance. The second localization estimates $\mathbf{m}_C$ for all passive devices C based on *estimated* distance differences given for all pairs of transmitters A, B with respect to C based on Section 6.1.2:

$$\hat{\tau}_{AB|C} \simeq \tau_{AB|C} = d_{AC} - d_{BC} = \|\mathbf{s}_A - \mathbf{m}_C\| - \|\mathbf{s}_B - \mathbf{m}_C\| \tag{6.12}$$

(note that $\hat{\tau}_{AB|C} = -\hat{\tau}_{BA|C}$), and the known (estimated) positions

$$\{\hat{\mathbf{s}}_A \mid A \in Transmitters\} \tag{6.13}$$

from the first localization.

6.2.1 Localizing transmitters

Our goal is to estimate the positions of speakers and microphones of transmitter devices based on their *estimated* distances from Equation (6.11).

Given estimated pairwise distances $\{\hat{d}_{AB} \mid A, B \in Transmitters\}$, we use a least-squares criterion to choose the *estimated* speaker and microphone coordinates of transmitters $\{\hat{\mathbf{s}}_A, \hat{\mathbf{m}}_A \mid A \in Transmitters\}$ so that it minimizes the following objective function:

$$\sum_{A,B \in \text{Transmitters}} \left(\hat{d}_{AB} - \|\hat{\mathbf{s}}_A - \hat{\mathbf{m}}_B\|\right)^2. \tag{6.14}$$

In general, the objective function of (6.14) is non-convex over the coordinates of speakers and microphones, making it prohibitively expensive to find the optimal solution. We use a closed-form solution given the constraints in (6.11) using existing work described in [47].

Due to approximation assumptions made by BeepBeep's distance estimation technique, BeepBeep does not have sufficient resolution to distinguish speakers and microphones within a phone. Therefore, we instead provide the input to the closed-form algorithm as follows: (1) we approximate all source-to-sensor distances across different devices using the estimated *device-to-device* distances given by BeepBeep, and (2) we input 0 for all source-to-sensor distances within a given device.

The closed-form solution for transmitter positioning assumes that the measured distances are precise, i.e.

$$\forall A, B \in Transmitters : \hat{d}_{AB} = \|\mathbf{s}_A - \mathbf{m}_B\|. \tag{6.15}$$

In practice, the measured distances are noisy, leading to inaccuracies in the computed transmitter coordinates. However, these coordinates can be improved using an iterative method based on existing work [64].

Iterative refinement

Given an initial solution, the method iterates using an auxiliary function for nonlinear least-squares optimization. We use the closed-form solution to initialize the iterative optimization of (6.14). The iterations are based on the following lemma:

Lemma 1. *Given a sequence of non-negative real numbers* $\{d_{nm}\}_{N\times M}$ *and two groups of points* $\{\mathbf{x}_1,\ldots,\mathbf{x}_M\}$ *and* $\{\mathbf{y}_1,\ldots,\mathbf{y}_N\}$ *in* $\mathbb{R}^2$ *or* $\mathbb{R}^3$*. Let* $\{\bar{\mathbf{x}}_1,\ldots,\bar{\mathbf{x}}_M\}$ *and* $\{\bar{\mathbf{y}}_1,\ldots,\bar{\mathbf{y}}_N\}$ *be given by*

$$\begin{aligned}\bar{\mathbf{x}}_m &= \frac{1}{N}\left(\sum_{n=1}^{N}\mathbf{y}_n + d_{nm}\frac{\mathbf{x}_m-\mathbf{y}_n}{\|\mathbf{x}_m-\mathbf{y}_n\|_2}\right)\\ \bar{\mathbf{y}}_n &= \frac{1}{M}\left(\sum_{m=1}^{M}\mathbf{x}_m - d_{nm}\frac{\mathbf{x}_m-\mathbf{y}_n}{\|\mathbf{x}_m-\mathbf{y}_n\|_2}\right)\end{aligned} \tag{6.16}$$

then

$$\sum_{m=1}^{M}\sum_{n=1}^{N}\left(d_{nm}-\|\bar{\mathbf{x}}_m-\bar{\mathbf{y}}_n\|_2\right)^2 \le \sum_{m=1}^{M}\sum_{n=1}^{N}\left(d_{nm}-\|\mathbf{x}_m-\mathbf{y}_n\|_2\right)^2. \tag{6.17}$$

Proof. See in [64]. □

Since values given in (6.17) are non-negative, the iteration (6.16) always converges to a local minimum. Depending on the starting coordinates, this local minimum may be far from the global one. Fortunately, as long as the noise is not "too" large and the closed-form solution is near the global minimum, the iterative method reaches a fixed point at the global minimum.

We apply Lemma 1 with the following substitutions that represent the estimated positions given by the closed-form method:

- $\{\mathbf{x}_1,\ldots,\mathbf{x}_M\} := \{\mathbf{m}_A^\circ \mid A \in Transmitters\}$
- $\{\mathbf{y}_1,\ldots,\mathbf{y}_N\} := \{\mathbf{s}_A^\circ \mid A \in Transmitters\}$

Let $\{\mathbf{s}_A^*,\mathbf{m}_A^* \mid A \in Transmitters\}$ be the convergence of (6.16) for the optimization (6.14). Then, we always have

$$\sum_{A,B\in Transmitters}\left(d_{AB}-\|\mathbf{m}_B^*-\mathbf{s}_A^*\|_2\right)^2 \le \sum_{A,B\in Transmitters}\left(d_{AB}-\|\mathbf{m}_B^\circ-\mathbf{s}_A^\circ\|_2\right)^2. \tag{6.18}$$

Equation (6.18) shows that the combination of the closed-form method and the iterative method provides a solution that is better than the closed-form method with respect to the objective function. Therefore, we use the output of closed-form+iterative.

Our Sonoloc implementation stops the refinement after 10 iterations. Sensitivity experiments suggest that additional iterations have negligible impact (see Section 7.2.9).

Sonoloc transmitter localization algorithm

A summary of the transmitter localization in Sonoloc is shown in Algorithm 2.

Input: Estimated pairwise distances: $\hat{D} = \{(A, B, \hat{d}_{AB}) \mid A, B \in Transmitters\}$.
$\forall A \in Transmitters : \hat{d}_{AA} = 0$.
Output: 2D Estimated positions: TrPos = $\{(A, \hat{\mathbf{s}}_A, \hat{\mathbf{m}}_A) \mid A \in Transmitters\}$.
TrPos = Apply closed-form solution [47] for $\hat{D}$;
TrPos = Iteratively apply update rule [64] for TrPos and $\hat{D}$;

Algorithm 2: Sonoloc transmitter localization.

6.2.2 Localizing passive devices

In Sonoloc, N speakers in N transmitters emit audio chirps, and these sounds are recorded by the N microphones in N transmitters and the M microphones in M passive devices. The speaker and microphone coordinates of transmitters are estimated using the closed-form (+iterative) algorithm described above. Next, we describe how Sonoloc estimates the coordinates of the M remaining microphones.

Based on the distance difference measurements described in Section 6.1.2 and the known coordinates of the transmitter speakers, the estimation of the position of a passive device's microphone maps precisely to the *source localization from range-difference measurements* problem studied in [79, 80, 86]. Crucially, the constraints corresponding to the different microphones are entirely *independent*, in that there is no dependence between microphones, their locations, recording, or computation. This property allows Sonoloc to scale. As long as microphones can record the sound samples with sufficient fidelity and undertake the procedure listed below, the system scales to an arbitrary number of passive devices, regardless of location. This property is useful because, e.g., in big lecture halls, hundreds of devices can be within audio range. In addition, the computation for each passive device is independent and can be executed in parallel.

For a passive device C, $\mathbf{m}_C$ can be estimated as follows. Given the estimated speaker positions of all transmitters $\{\hat{\mathbf{s}}_A \mid A \in Transmitters\}$ and the observed distance differences $\{\hat{\tau}_{AB|C} \mid A, B \in Transmitters\}$, we choose the *estimated* position $\hat{\mathbf{m}}_C$ that minimizes

$$\sum_{A,B \in \text{Transmitters}} \left[\hat{\tau}_{AB|C} - \left(\|\hat{\mathbf{s}}_A - \hat{\mathbf{m}}_C\| - \|\hat{\mathbf{s}}_B - \hat{\mathbf{m}}_C\|\right)\right]^2. \tag{6.19}$$

The optimization (6.19) has a non-convex cost function of only two coordinates of $\hat{\mathbf{m}}_C$. Thus, (6.19) is simpler than (6.14), and the existing closed-form solutions work well for the optimization (6.19). Sonoloc uses the closed-form solutions described in [80] to solve the optimization (6.19). Note that [80] describes localizing *speakers* based on known microphone locations, but in Sonoloc, we localize microphones based on known speaker positions. Our

problem is symmetric in the sense that distance differences can be defined between pairs of speakers and one microphone as we do, or, between pairs of microphones and one speaker as described in [80].

Iterative refinement

As before, the closed-form solution assumes that the measured distance differences are accurate. In practice, the differences are noisy, leading to inaccurate output. In order to improve the quality of the solution given by the closed-form method, we use an iterative algorithm based on prior work [64]. The iteration is based on the following observation:

$$\begin{aligned}\left[\tau_{n_1 n_2|m} - \left(\|\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_1}\|_2 - \|\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_2}\|_2\right)\right]^2 \leq\ & 2\left[\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_1} - \frac{\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_1}}{\|\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_1}\|_2}\mu_{n_1,n_2|m}\right]^2 \\ & + 2\left[\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_2} - \frac{\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_2}}{\|\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_2}\|_2}\eta_{n_1,n_2|m}\right]^2\end{aligned} \tag{6.20}$$

where

$$\begin{aligned}\mu_{n_1,n_2|m} &= \frac{1}{2}\left(\|\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_1}\|_2 + \|\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_2}\|_2 + \tau_{n_1,n_2|m}\right) \\ \eta_{n_1,n_2|m} &= \frac{1}{2}\left(\|\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_1}\|_2 + \|\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_2}\|_2 - \tau_{n_1,n_2|m}\right).\end{aligned}$$

The equality of (6.20) holds if and only if

$$\hat{\mathbf{m}}_m = \frac{1}{2}\left(\hat{\mathbf{s}}_{n_1} + \frac{\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_1}}{\|\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_1}\|_2}\mu_{n_1,n_2|m} + \hat{\mathbf{s}}_{n_2} + \frac{\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_2}}{\|\hat{\mathbf{m}}_m - \hat{\mathbf{s}}_{n_2}\|_2}\eta_{n_1,n_2|m}\right). \tag{6.21}$$

Using the technique called "auxiliary function for nonlinear least-squares optimization", (6.20) and (6.21) imply that for all values n_1, n_2 ($n_1 < n_2$), the cost function given by (6.19) is decreased when we use the right hand side of (6.21) to update $\hat{\mathbf{m}}_m$.

Since the work in [64] shows that this solution is never worse than the closed-form solution, we always use the coordinates provided by the iteratively refined solution.

Our implementation stops the refinement after a maximum of 2000 iterations or when the improvement in the value of the objective function falls below a threshold where Sonoloc's overall accuracy is not affected anymore. Our sensitivity experiments suggest that the refinement is required for good accuracy; no more than 2000 iterations were needed in any of our experiments (see Section 7.2.9).

Sonoloc passive device localization algorithm

A summary of the passive device localization in Sonoloc is shown in Algorithm 3.

Input: Estimated distance differences:
$\hat{D}_{TDOA} = \{(A, B, C, \hat{\tau}_{AB|C}) \mid A, B \in Transmitters \wedge C \in Passive\}$.
Input: 2D Estimated positions: $\{\hat{\mathbf{s}}_A \mid A \in Transmitters\}$
Output: 2D Estimated positions: PassivePos = $\{(A, \hat{\mathbf{m}}_A) \mid A \in Passive\}$.
PassivePos = Apply closed-form solution [80] for $\hat{D}_{TDOA}$;
PassivePos = Iteratively apply update rule [64] for PassivePos and $\hat{D}_{TDOA}$;

Algorithm 3: Sonoloc passive device localization.

6.2.3 Sonoloc localization algorithm

A summary of the complete Sonoloc localization algorithm is shown in Figure 6.2.

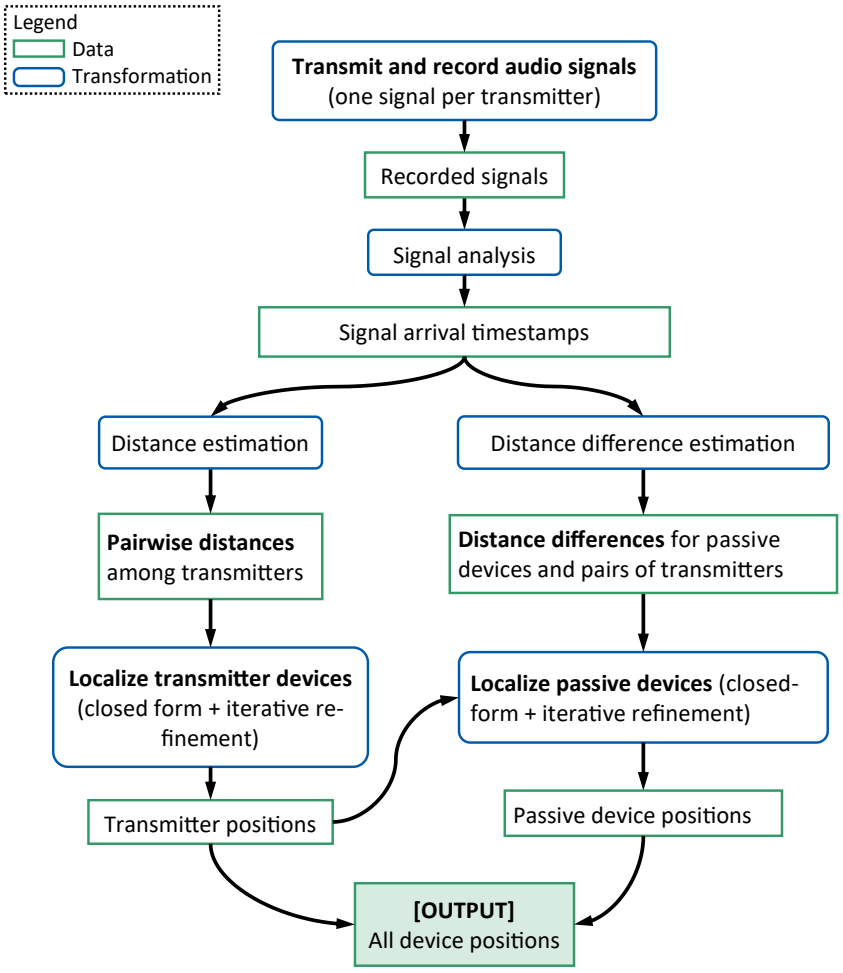

Figure 6.2: Complete Sonoloc localization algorithm

Chapter 7

Evaluation of Sonoloc

In this chapter, we present our evaluation of Sonoloc. First, Section 7.1 describes our prototype implementation. Then, in Section 7.2, we present the results of an extensive experimental evaluation that includes both simulations and real-world testing.

7.1 Prototype implementation

In order to conveniently conduct many experiments under controlled conditions, and to experiment with different algorithms on the same recorded data, we chose a server-based implementation. Sonoloc apps installed on participating devices send their recorded sound samples to the server, which analyzes the signals, computes and disseminates the devices' relative position map. All communication with the server happens over the internet (in our implementation, over Wi-Fi).

7.1.1 Protocol coordination

A localization protocol session may be initiated by any of the participants. The initiating user's device (called the *initiator*) sends a message to the server, to which the server responds by issuing a session ID. Then, the initiator advertises the new session, with the session ID and the server's address, to nearby devices via Bluetooth Low Energy [78]. Nearby devices scan continuously on Bluetooth Low Energy for such advertisements. Upon receiving a Sonoloc session announcement, devices register with the server for that session.

Then, the initiator informs the server that the registration phase is over, and it is time to start transmitting signals. This notification can be triggered either after a specified amount of time, or, since the user can query the server for the number of registered participants, by user action. The server chooses an initial, random set of transmitters, and decides on a transmission order for them. First, it asks all devices to start saving audio samples, then it asks the next transmitter to transmit the signal, then, after a delay that allows the echoes to fade, it asks all devices to stop saving audio samples, and upload what they have recorded. When all recorded signals are

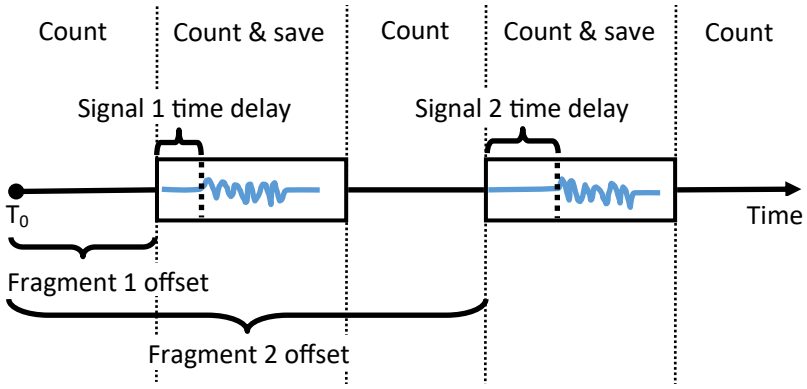

Figure 7.1: Sonoloc audio recording overview. T_0 represents the start of the recording at the beginning of the session. The arrival time of a signal relative to T_0 is the sum of the corresponding fragment's offset (given by sample counting) and the time delay (given by time delay estimation). Note that the time difference of arrival between two signals can be computed without knowing T_0's global timestamp.

in from the initial transmission set, the server analyzes the signals, positions all devices, and chooses a new transmitter based on the map. Then it starts another transmission sequence for the new transmitter. Then, it again analyzes the (new) signals, positions all devices, this time considering the data from the new transmitter. This iteration goes on until the specified number of transmitters is reached.

7.1.2 Overview of audio I/O in Sonoloc

In order to make sure that the recorded audio data is of good quality, Sonoloc uses a 48 kHz sampling rate for both transmission and recording, and each audio sample is encoded in 16 bits. At development time, this was the highest-quality audio format that our devices could reliably handle. Assuming no audio compression, these settings generate 96000 bytes per second on a single microphone. Regardless of hardware support, a higher-quality setting would generate more data that needs to be processed and optionally uploaded, revealing a tradeoff between audio quality and resource consumption.

For the localization algorithms to work, devices must be able to count the number of audio samples (an indicator of elapsed time) between any pair of signals. The simplest way to do this is to record continuously and save all incoming audio samples during the entire session. A more efficient way is to save the incoming audio samples only when there is a transmission to be expected (controlled by the server as described below), and only *count*, but not save, the audio samples otherwise. This optimization significantly shortens the recorded waveforms on which signal detection must be performed. It also establishes the invariant that each "fragment" (continuous period of recorded and saved samples) contains a single signal instance (and its reflections), which simplifies the signal analysis. The offsets between such fragments are tracked via continuous sample counting. This is Sonoloc's approach, and it is summarized in Figure 7.1. In this approach, devices still *count* audio samples during the entire session, but this

is not necessary. As a further optimization, it would be sufficient to turn on the microphone and start to count samples right before the first signal is expected to arrive.

7.1.3 Multi-channel transmission and recording

If we transmit the same signal on multiple speakers, the two signals would arrive at slightly different times at the receiver, and the receiver would find it difficult to distinguish them from each other. In order to avoid this problem, Sonoloc configures the speakers for stereo output, transmits the signal on one of the speakers (the top speaker on the Motorola Nexus 6), and mutes the other speaker.

The case of audio recording is more complicated. Many recent smartphones have at least two microphones built into them, and, in an attempt to improve robustness, it may be tempting to combine signal arrival timestamps across multiple microphones. However, since the microphones of a device are in different locations and these locations change with the device's orientation, it is not straightforward to do so. An alternative approach would be to choose a single microphone across *all* of Sonoloc's computations. However, this approach is inflexible: it does not use information from "non-chosen" microphones even if they correctly detected a signal that the chosen microphone could not detect. Such a situation may occur if, for example, a microphone is obstructed by the device owner's hand or by the phone's body itself.

Thus, Sonoloc records signals on all available microphones, and, each time a distance or distance difference needs to be estimated, for each relevant signal, Sonoloc dynamically chooses one microphone to use. Sonoloc uses two different microphone selection strategies for determining distances and distance differences. In the following, we describe our solution for these cases separately.

Microphone selection for distance estimation

For a transmitter A, estimating the distance from another transmitter B requires estimating the time difference of arrival of its own signal and the signal from B. In order to make sure that the observed time difference of arrival is not affected by the different positions of the receiving microphones, the corresponding timestamps must originate from the same microphone. Sonoloc chooses the microphone that reported the earliest signal arrival timestamp for B's signal. Then, it uses the same microphone for A's own, easier-to-detect signal. Thereby, Sonoloc can take advantage of the fact that one of A's microphones may have an unobstructed line of sight to B while other microphones do not.

Using different microphones on the same device for estimating distances to different devices adds noise to distance estimates. We accept this extra error since using the *best* microphone for a given measurement proves to be necessary in practice. Consider an alternative where a single microphone, however chosen, is used on a device. Unfortunately, if this microphone cannot be used for distance measurements from even a single transmitter (which can happen due to

physical barriers or other environmental factors), then we receive no distance estimate for this pair. One option is to ignore this measurement. However, since the transmitter localization method we use requires *all* pairwise distances, doing so would cause a single poor measurement to result in the transmitter being discarded by all devices!

Microphone selection for distance difference estimation

For a passive device C to estimate the distance difference from two transmitters A and B, C needs to estimate the TDOA of those signals. In order to make sure that the observed time difference of arrival is not affected by the different positions of the receiving microphones, the corresponding timestamps must originate from the same microphone. Now let Ref denote the reference transmitter, which could be any of the transmitters. Since the positioning of C requires the estimation of the TDOAs for all transmitter pairs in $\{(TX, Ref) | TX \in Transmitters \wedge Ref \neq TX\}$, effectively, C must use the same microphone for all audio signals.

Sonoloc uses the *majority microphone* of C for this purpose. The majority microphone is the one that detected a signal from the largest number of the transmitters. To estimate the distance difference for a specific pair of transmitters, the majority microphone is used. If, for at least one transmitter of the pair, the majority microphone did not detect a signal, then Sonoloc reports that the distance difference cannot be determined, thus it will not be considered by the passive device localization algorithm as a constraint.

In contrast to transmitter localization, passive device localization can work with any (sufficiently large) number of distance differences. Thus, it is less risky to choose a single microphone (e.g., the one that provides the best signal quality for the majority of measurements) and ignore an individual distance difference whenever the majority microphone cannot be used. When a distance difference is ignored, we still lose some accuracy, but, unlike losing a transmitter, which affects *all* passive devices, only a single passive device is affected. Since the associated risk is smaller, Sonoloc uses the majority microphone approach for passive device localization.

7.1.4 Tools used

The Sonoloc prototype app was implemented on Android 7.1.1. The server application, written in Java and MATLAB, runs on top of Oracle JDK 8 and uses MATLAB Compiler SDK 6.4 to interface with MATLAB implementations of the localization algorithms. The source code uses Java 8 features extensively, and it relies on some Apache Commons libraries. For source code development, we used IntelliJ IDEA 2017, Android Studio 3.0, and MATLAB R2017b. For visualizing our results, we used MATLAB R2017b.

Noise type	Lowest robust SNR	trials
Synthetic Gaussian	1.53 dB	137500
Real noise (Cafeteria)	−22.6 dB	1872700
Real noise (Talk)	−24.5 dB	161600
Real noise (Street)	−34.1 dB	112200
Real noise (Poster session)	−24.2 dB	106600

Table 7.1: SNR levels for reliable signal detection

7.2 Experimental evaluation

In this section, we present results of our experimental evaluation of Sonoloc. The evaluation has three parts. First, we evaluate the performance of Sonoloc's signal detection under different types of noise and signal-to-noise ratios. Second, we present results of extensive simulations that allow us to explore many more device layouts than we could reasonably cover in real experiments. It also enables us to explore the impact of factors like the selection and number of transmitters, inaccuracies in detected signal arrival timestamps, and z-offsets. Finally, we present real-world experiments in different rooms with a prototype implementation running on up to 15 Motorola Nexus 6 smartphones [27] and up to 100 Raspberry Pi 3 Model B units [23] as passive devices.

7.2.1 Signal detection

Detecting the audio chirp signal in a recorded waveform is essential for accurate localization. We evaluate the Sonoloc signal detector under controlled conditions by mixing the clean reference signal with various types of noise at different signal-to-noise ratios (SNR). We are using synthetic and recorded noise for our evaluation.

For synthetic noise, we use Gaussian random noise with a zero mean and unit standard deviation. For real noise evaluations, we recorded background noise in various environments, including street noise from an open window, a talk (presentation), a poster session, and a cafeteria during lunch hour. The recorded noise recordings are several minutes long; in each experiment, we mix in the signal at a different random offset within the noise recording. During mixing, the amplitude of the noise is adjusted to achieve a desired SNR.

We run the signal detector on the noisy recordings and compare the detected signal position with the known ground-truth position of the signal in the recording. Table 7.1 shows the minimal SNR required for reliable detection with different types of noise.

For each noise type, we initialized the noise generator with a high noise level, and evaluated the detector's performance in 100 trials. If any detection instance failed or if the estimated offset had more than 3 samples of error, then we divided the noise amplitude by 1.2. (We determined the latter value empirically in order to provide a good balance between the SNR resolution and the computation time of our evaluation script.) Otherwise, we let it run for another 100 trials.

After successfully completing about 100000 total trials, we concluded that the detection is robust at the current SNR level. Detection error is defined as the difference between the estimated and true signal position, in number of audio samples.

As we can see, Sonoloc's detector can reliably detect signals in real noise down to SNRs below −20 dB. (For reference, given a sampling rate of 48 kHz, an error of one sample corresponds to a distance of 7.15 mm, the distance sound travels (at 20 °C in dry air) during one sample period.) This robustness is possible because the acoustic spectrum of real noise is typically limited. Such a spectrum contains strong frequencies affected by loud noise, and some weak frequencies largely unaffected by the noise. Given such a situation, the whitening filter selectively enhances the weaker frequency bands that are unaffected by the noise. In contrast to realistic noise, synthetic Gaussian noise is full-band, so the technique we just described does not help.

7.2.2 Evaluation metric

We evaluate the positioning accuracy in terms of the *interpoint distance error (IPDE)*: for each pair of devices, the absolute difference between the true distance and the distance based on the estimated positions between the devices' centers, in meters. We consider a device's center the average of the coordinates of its speaker and microphones. Assuming that a coordinate is a struct containing one field per dimension ($c = (x, y, z)$), we define the average of a collection of coordinates as follows:

$$CoordAvg(C : Collection < Coordinate >) := \frac{1}{|C|} \begin{bmatrix} \sum_{c \in C} c.x \\ \sum_{c \in C} c.y \\ \sum_{c \in C} c.z \end{bmatrix} \tag{7.1}$$

7.2.3 Positioning simulations

Next, we show results of our extensive simulations to explore Sonoloc's accuracy in positioning devices systematically under many different device layouts, transmitter selections, various levels of signal detection inaccuracy, z-offsets, etc. We begin with a description of how we perform our simulations.

Device layouts We simulated a large number of room shapes and device layouts likely to arise in practice. In all cases, devices are placed in a plane and minimally 50 cm apart from each other, reflecting a typical private space people tend to keep.[1]

Random: Pick random coordinates for the devices within either a 20 m × 20 m quadratic space or a circular space with a 10 m radius.

[1]People can stand closer in crowded circumstances like a subway train, but it is not clear if Sonoloc would normally be used in this situation.

Clustered: Pick a given number (e.g. 5) of cluster centroids at a random location within a 20 m × 20 m space at least 1 m apart. Then, populate each cluster with 4 to 30 additional devices at random locations with a distance from the centroid exponentially distributed with a mean of 1 m.

In the following regular device layouts, we perturb each device's nominal location by choosing a uniformly random 2D position up to 10 cm away from the nominal position to account for typical variations in device positions relative the users' nominal position (e.g. in a given seat).

Concentric circles: 100 devices are laid out in two concentric circles with radii 9 and 18 m, respectively.

Cafeteria: Devices are arranged around 36 rectangular tables with 8 devices each. Table centers are arranged in a grid 3 m apart.

Banquet: Devices are arranged around 10 round tables with 12 devices each. Tables are placed at random locations within a 25 m × 25 m room.

Small restaurant: Devices are arranged around 6 rectangular tables 2.5 m from each other with 6 devices each in a small 5 m × 10 m rectangular room.

Lecture hall: Devices are placed randomly into 20 rows of 20 seats each with a separation of 0.5 m between rows and seats.

Conference room: Devices are placed on a long, rectangular table, where seats are separated by 0.5 m and the two long edges are 2 m apart.

Generating device positions We place the phones' speakers and microphones in the following way. First, the speakers of the devices are laid out according to the device layout specification described above. Then, since Sonoloc positions devices in a 2D plane, but in practice, devices may be carried at different levels above ground (in hand while standing, in pocket, on table), we simulate device offsets in the Z dimension according to the specification given in the configuration. These Z offsets are added to the coordinates of the speakers. Finally, we assign coordinates to each microphone by adding a randomly chosen 3D vector to the speaker coordinate. The length of this vector depends on the speaker-microphone distance specified in the configuration.

The generation of device positions accepts the following simulation parameters:

- **Z-offset:** by default, we assume that all devices (more precisely, their speakers) are located in the same 2D plane (Z offset = 0).
- **Speaker-microphone distance:** by default, this parameter is set to the known speaker-microphone distance of the device.

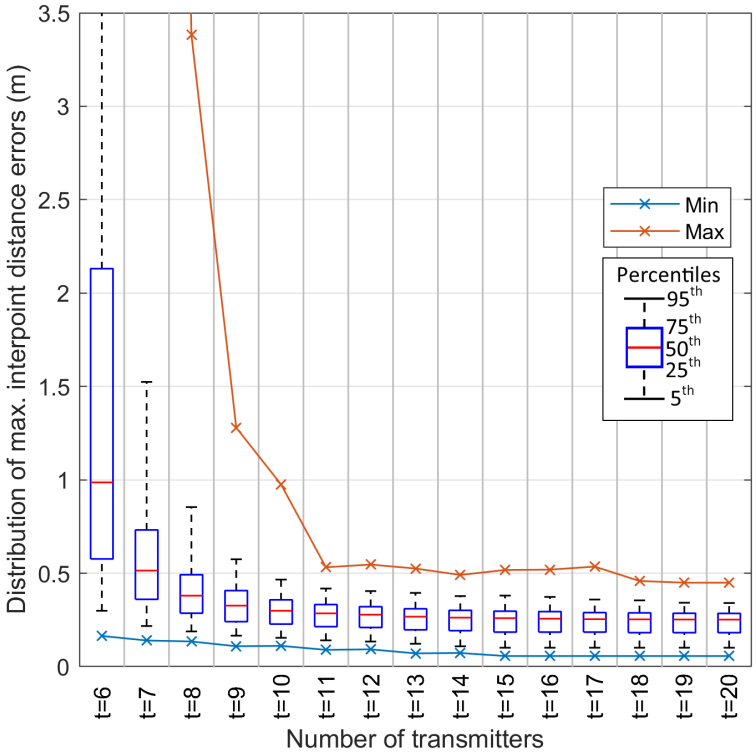

Figure 7.2: Max IPDE vs. #transmitters; 500 scenarios. Percentiles 5, 25, 50, 75, and 95 shown.

Simulating signal propagation There is no signal processing in these simulations. Instead, to simulate the effect of signal detection errors, we perturb the actual arrival time according to the distribution given in the configuration. The simulation parameters related to signal propagation are the following:

- **Signal processing inaccuracy:** by default, this is a uniform random number of audio samples in the range $[-1, 15]$.
- **Air temperature for signal propagation simulation:** by default, we set it to 20 °C.

Localization parameters used in simulations

- **Initial number of transmitters:** by default, we start with 6 randomly chosen transmitters
- **Final number of transmitters:** by default, we iteratively add transmitters up to 13 transmitters.
- **Air temperature for localization:** by default, the localization algorithm uses the same air temperature as the one used for simulating the signal propagation. This parameter determines the speed of sound that the localization algorithm assumes.

7.2.4 Overall positioning accuracy

Figure 7.2 shows the statistic of the maximal interpoint distance error (IPDE) in each individual experiment as a function of the number of transmitters used, over 500 simulated sample runs of the Sonoloc protocol. Each run used a different device layout chosen in equal proportions from the types described above. The results show that with 11 or more transmitters, the maximal IPDE is around 0.5 m, and, in half of the experiments (see median), the max. IPDE is below 0.3 m. Considering conservative assumptions in our simulations (challenging device layouts, inaccurate signal detection, etc.), this is a very strong result.

7.2.5 Simulated sensitivity experiments

We have performed extensive simulations to explore the sensitivity of Sonoloc's positioning accuracy to various external factors (ones that the system designer cannot control). When exploring a given parameter P, we do the following. Given a list of parameter values we want to explore (e.g., $P = \{1, 2, 3\}$), we generate a configuration for each value in the list. In these configurations, we use the default values for all other parameters.

Then, for each configuration, we generate all (13) device layout types once, and, for each device layout, we generate 10 random initial 6-transmitter sets. Therefore, in total, we have 130 rounds per configuration. For instance, a parameter group for exploring 3 values of P would look like this:

- config1: P=1, all others default: 13x10 rounds
- config2: P=2, all others default: 13x10 rounds
- config3: P=3, all others default: 13x10 rounds

For each round, we apply the simulated effects of the signal processing inaccuracy again.

In the following, we present our results. These results report the statistics of *all individual IPDEs*, not just the statistics of the max. IPDEs per experiment.

Signal detection inaccuracy

As discussed above, we assume uniform random simulated signal detection errors in the range $[-1, 15]$ samples, as this choice yields the best agreement with our empirical results. To understand the sensitivity, we also explored other error distributions. Assuming perfectly accurate signal detection, the median and maximal IPDE is below 0.06 m and 0.33 m, respectively. Note that there are errors even with perfect signal detection ($e = 0$), because the devices' orientations affect their positioning in the 2D plane. Assuming large detection errors in the range $[-25, 25]$, the median and maximal IPDE increases to 0.11 m and 1.55 m, respectively, and only 73 out of 130 trials succeeded in localizing the devices. Figure 7.3 shows the distributions of interpoint distance errors for different levels of signal detection inaccuracy (e). We conclude that Sonoloc can tolerate frequent but modest signal detection errors well.

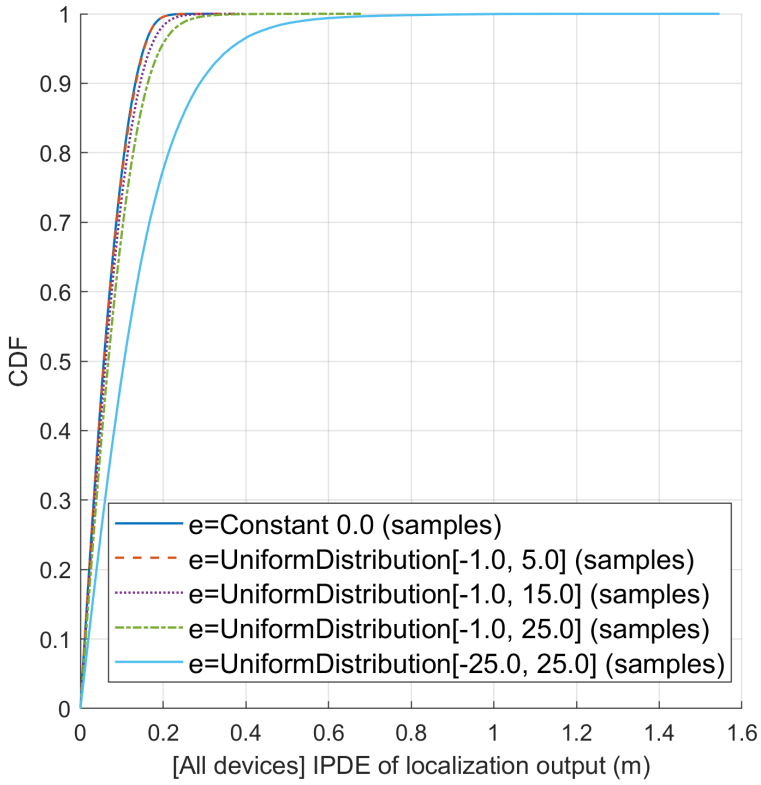

Figure 7.3: Positioning accuracy vs. signal detection error distribution e (samples).

Z-offsets

Sonoloc does 2D localization, but in practice, devices may have vertical offsets. To explore the impact of such offsets on Sonoloc's positioning accuracy, we assigned each device a uniform random z-offset. For a z-offset in the range $[-0.5, 0.5]$ m, the median IPDE is unaffected and the maximal IPDE increases to 0.87 m. However, for z-offsets in the range $[-1, 1]$ m, the median and maximal IPDEs increase to 0.1 m and 1.95 m. Details can be seen on Figure 7.4. We conclude that Sonoloc can tolerate z-offsets of up to 0.5 m with acceptable accuracy.

Ambient temperature

The speed of sound varies with air temperature. In our simulations, Sonoloc's calculations assume a temperature of 20 °C, and our simulations show that Sonoloc can tolerate temperature deviations of up to 5 °C with modest impact on IPDEs. However, larger temperature deviations require compensation. Future smart devices will likely have thermometers, making it possible to perform this compensation automatically. In our experience, if the true ambient temperature is different from what the localization algorithm expects, the estimated position map will be a scaled (either larger or smaller) version of the true position map, but the *shape* of the map and the relative positions of devices remain unaffected.

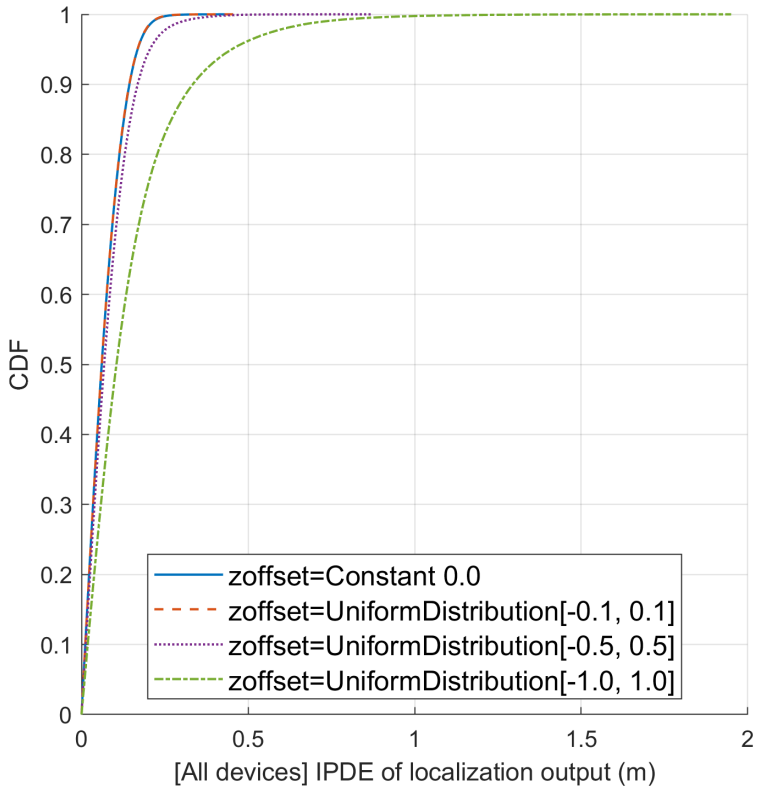

Figure 7.4: Positioning accuracy vs. Z offset.

Device layout

Figure 7.2 includes results with all different device layout types. We also ran separate simulations with the individual device layout types described above. In general, device layouts can be more or less challenging, depending on the extent to which they constrain transmitter selection. The easiest layouts are the small random layouts (15 devices), followed by a small clustering layout (3 clusters, 15 devices), then the conference room (30 devices) and small restaurant layout (30 devices). The most challenging are the large random layouts (100 devices), the lecture hall (100 devices), and the cafeteria layout (288 devices). The latter are responsible for the largest IPDEs shown in Figure 7.2.

Speaker-microphone distance

We compared the localization accuracy for different distances between the speaker and microphone of the same device. (2 microphones are generated per device, both with the same speaker-microphone distance.) Figure 7.5 show how the accuracy depends on the speaker-microphone distance. The figure shows that a smaller speaker-microphone distance tends to produce better accuracy. The reason for this result is that the speaker-microphone distance affects the distance estimation accuracy, and the inaccuracy of distance measurements affects all subsequent steps of the localization including passive device localization. According to the figure, about 14 cm speaker-microphone distance can give acceptable results, but increasing it

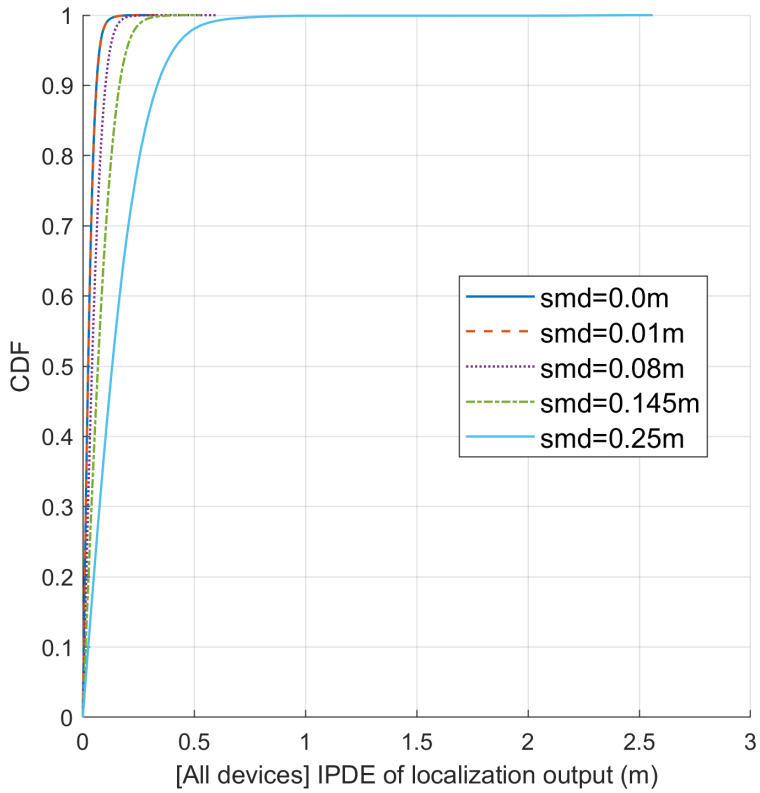

Figure 7.5: Positioning accuracy vs. speaker-microphone distance smd (m).

to 25 cm can result in IPDEs larger than 2.5 m.

Number of devices

In order to evaluate Sonoloc's scalability, we performed simulations for different numbers of devices up to 500 devices. Since many of our simulated device layout types only support a given number of devices, in this simulation, we instead generated random device layouts with n devices in a 20 m × 20 m square space. Figure 7.6 shows the result. The figure shows clearly that Sonoloc can position a large number of passive devices with high accuracy.

7.2.6 Live experiments

We conducted live experiments with our Sonoloc prototype running on Motorola Nexus 6 phones [27]. To enable experiments with large numbers of devices at reasonable cost, we also ran Sonoloc on Raspberry Pi (RPi) single-board computers [23] with Plantronics Audio 300 microphones [7] and Sabrent AU-MMSA USB audio adapters [73]. According to our measurement, the distances between the speaker and the two microphones on these phones are 14.5 cm and 1 cm. We experimented in different rooms and several buildings on our campus, and under different noise conditions.

The noise conditions were *quiet*: only background noise from HVAC and other equipment (38 dB–40 dB); *speech*: playback of recorded speech (49 dB–55 dB); and *music*: playback of

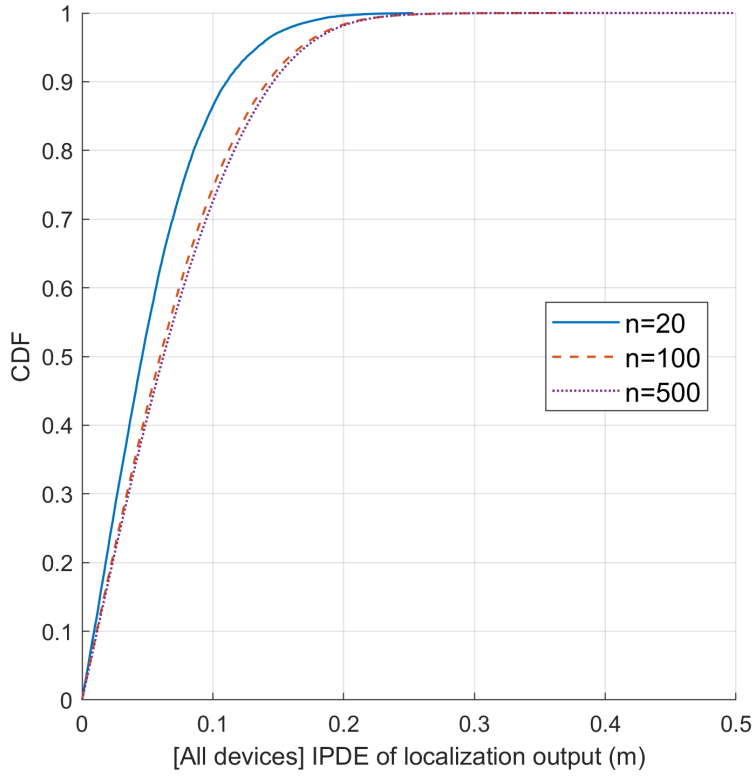

Figure 7.6: Positioning accuracy vs. #devices n.

recorded music (50 dB–62 dB). (Measured with a PeakTech 5055 sound level meter [57] with A-weighting.) The background speech and music were replayed using a pair of 150 W JBL Control 1 Pro speakers [68] connected to a 2x75 W Alesis RA150 amplifier [53], except in the large lecture hall. There, we used the existing room audio system consisting of a DYNACORD DSA 8405|8410|8805 (3-component) amplifier [21] connected to four JBL CBT 100LA line array column loudspeakers [67] at the front of the lecture hall, and 25 JBL Control 24C/CT ceiling speakers [69]. The phones emitted chirps at the highest possible volume setting using only one channel. During the experiments, the phones and RPis were not moved.

Our first set of experiments were conducted with up to 15 phones in a small 10-seat conference room, a long but narrow 28-seat conference room, a medium-sized 60-seat classroom, a 100-seat classroom, and a 180-seat classroom. In each case, we distributed phones throughout the rooms. The phones were held above the seats and tables using smartphone holders. There was an unobstructed line of sight between most but not all pairs of phones. Pairwise distances among phones ranged from 0.7 m to 2.7 m in the 10-seat room, from 0.5 m to 8.6 m in the 28-seat room, from 0.6 m to 4.3 m in the 60-seat classroom, from 0.9 m to 9.6 m in the 100-seat classroom, and from 0.6 m to 13.1 m in the 180-seat classroom.

In each case, we conducted experiments with silence, speech, or music replayed. Figure 7.7 shows the statistic of the maximal IPDE in each individual experiment as a function of the number of transmitters used across 36 protocol runs in the different rooms, and different background noise. The results show that Sonoloc achieves maximal IPDEs of less than 0.5 m for 10 or

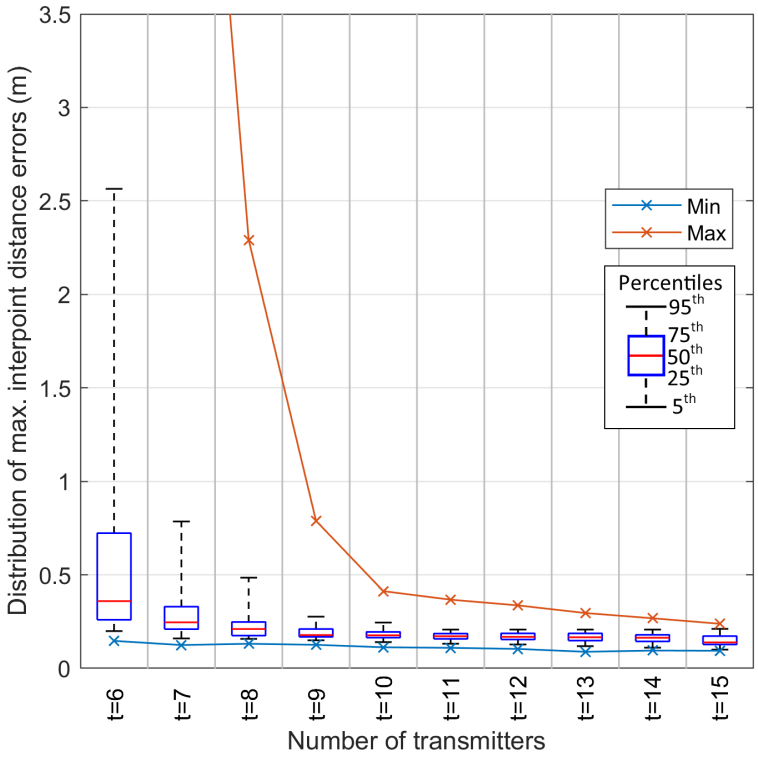

Figure 7.7: Max IPDE vs. #transmitters; various rooms; up to 15 phones. Percentiles 5, 25, 50, 75, and 95 shown.

more transmitters. The presence of background noise did not significantly affect the IPDEs (not shown).

We also conducted experiments in a large, 180-seat lecture hall with 15 phones and 100 RPis. In the lecture hall, the pairwise distances between devices range from 0.5 m to 17.8 m. Since the RPis have no speakers, they cannot be chosen as transmitters. Therefore, we conducted the experiment as follows. We randomly chose 115 out of the 180 seats in the large lecture hall to be "occupied". We ran a simulation with the exact room layout and seat occupancy to let Sonoloc choose a set of transmitters from the 115 occupied seats. Then we placed the phones into the seats that were chosen as transmitters, we filled the remaining seats with RPis, and ran the protocol as normal, except that we constrained the transmitter selection to within the set chosen in the simulation. Figure 7.8 shows a photo of the experiment setup.

Figure 7.9 shows the statistic of maximal IPDEs with 115 devices in the lecture hall, as a function of the number of transmitters used. The results include seven protocol runs with different types of background noise. The results show that Sonoloc can position 115 devices to a maximal IPDE of 0.5 m with only 15 audio chirps under real-world conditions. Again, the presence of background noise did not significantly affect the IPDEs (not shown).

Compared to the results shown in Figure 7.7, where only phones were used, Sonoloc requires 15 transmitters to achieve a maximal IPDE under 0.5 m in this experiment, and the median and average IPDE are higher. Further investigation showed that the higher IPDEs are associated with the position of the RPis and not the phones. These devices exhibit a larger sampling

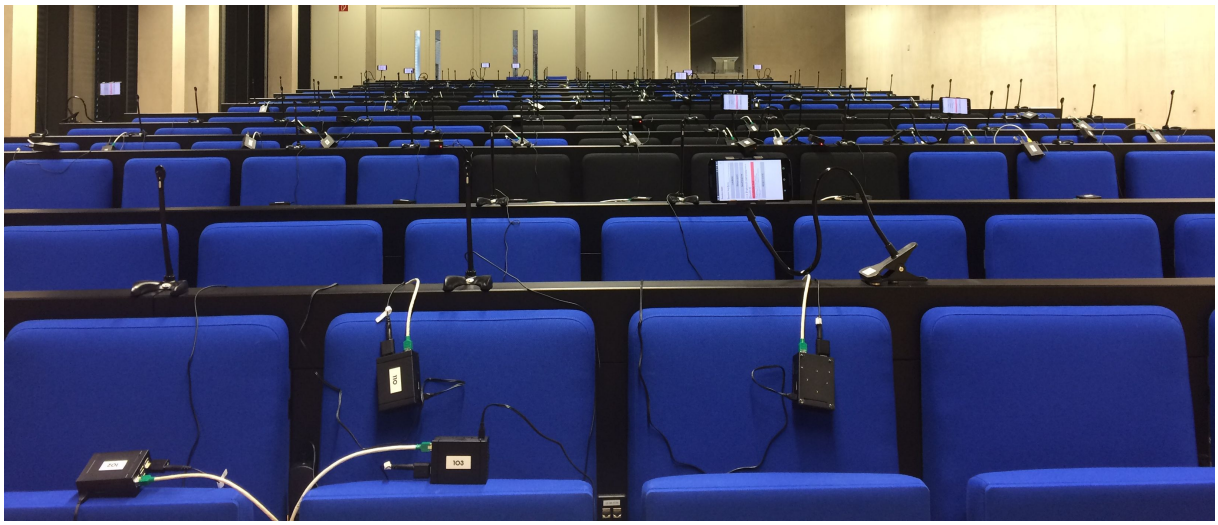

Figure 7.8: Photo of the RPi experiment setup in a lecture hall

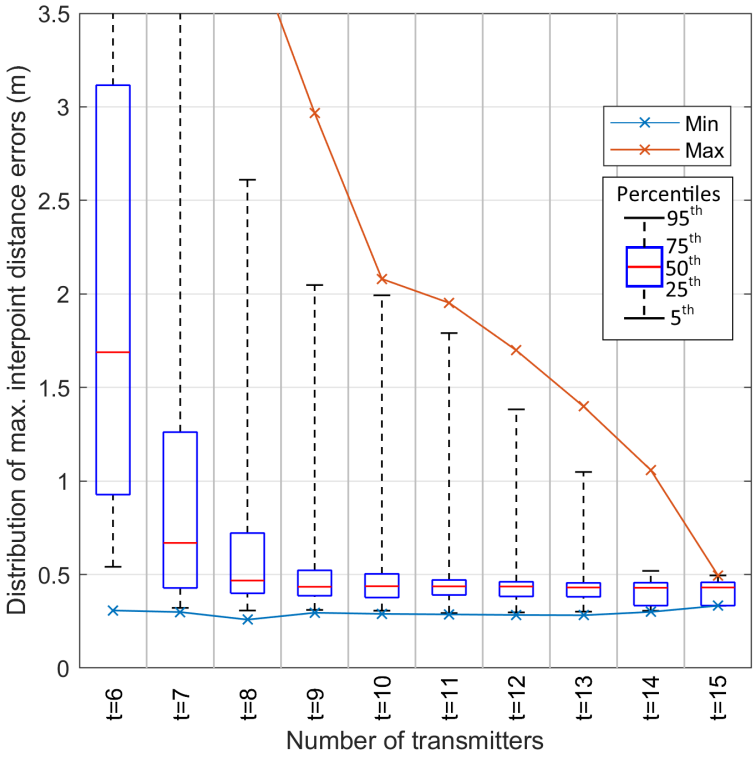

Figure 7.9: Max IPDE vs. #transmitters; 180-seat lecture hall; 115 devices. Percentiles 5, 25, 50, 75, and 95 shown.

rate mismatch, most likely due to the lower quality of the USB sound cards we used with them. These additional errors would not occur in practice, where all devices are assumed to be smartphones. Additionally, as we point out in Section 7.2.8, the effect of the sampling rate mismatch increases as the time gap between signal transmissions increases. In our real experiments, the transmissions were separated by a larger time gap than what could be achieved in an optimized system, which amplified the effect of the sampling rate mismatch. Nevertheless, we think the accuracy is still very good, and perfectly adequate for the application requirements Sonoloc seeks to meet.

Since each passive device localization is completely independent of other passive devices, if all other factors are equal, the number of transmitters does not and cannot depend on the number of passive devices. If the number of passive devices is large, some passive devices can act as obstacles between certain transmitter-receiver pairs, but, in practice, it seems unlikely that devices align in a way that many devices simultaneously block the line of sight between a pair of devices.

Operational range: Sonoloc's range is limited by background noise, audio signal attenuation, and the properties of speakers and microphones. In our experiments, Sonoloc worked reliably at distances of up to 18 m.

Figure 7.10 shows an example of a positioning map with true and estimated positions. The corresponding experiment took place in a large lecture hall with background music. The experiment included 100 RPis and 15 phones.

7.2.7 Termination condition

Sonoloc continues to choose transmitters iteratively until a fixed number of transmitters is reached. Our evaluation suggests that 15 transmitters are sufficient in practice. One might argue that a more sophisticated termination condition would allow the protocol to determine the optimal number of transmitters on a case-by-case basis. A good termination condition should ensure good accuracy, and, for practical reasons, it should provide a bound on the number of transmitters. In the following, we discuss two potential approaches.

Termination based on consistency with input data: With this approach, we stop adding transmitters when the discrepancy between the position map and the input distances/distance differences drops below a certain threshold. However, the input distances/distance differences will inevitably contain some errors, and it is not clear whether consistency with this erroneous data yields a useful termination condition.

Termination when the map stops changing: With this approach, we stop adding transmitters when, according to some metric, the change in the position map goes below a given threshold. There are two issues with this approach. First, when a new transmitter with inaccurate signal analysis results is added, the position map may change slightly for

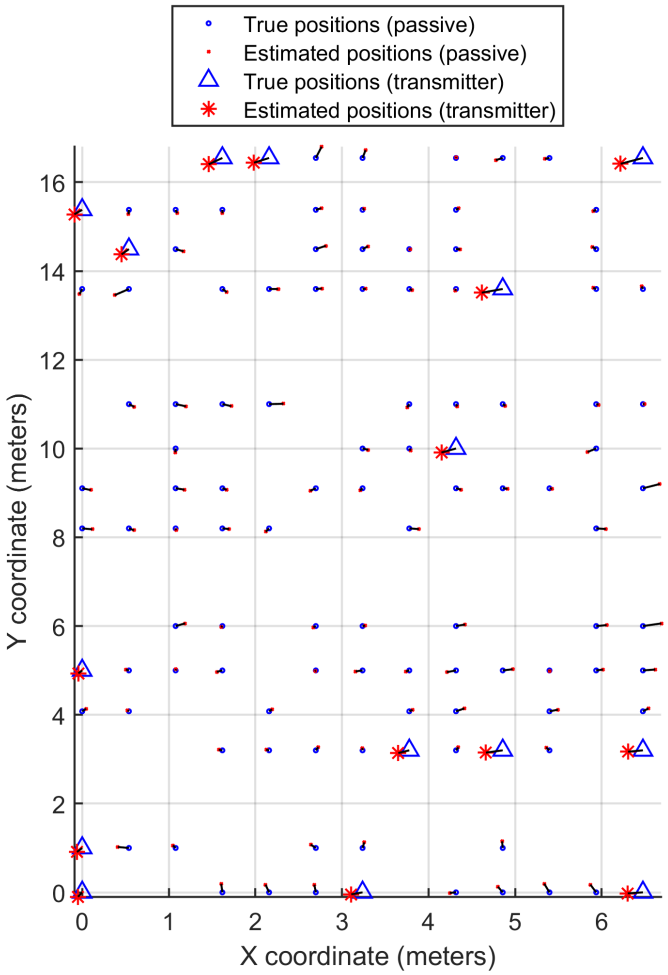

Figure 7.10: Example positioning map with true and estimated positions

the worse, which may prevent the termination of the protocol, and lead to too many transmitters. Second, adding a transmitter does not guarantee a significant change in the map, which may cause the protocol to terminate early regardless of the accuracy of the current map.

The problem with these more sophisticated termination conditions is that Sonoloc only knows the (noisy) input distances and distance differences. Without any information about the ground truth positions, it is not possible to determine whether the current position map is accurate enough. Although we experimented with such termination conditions, we found that the best-performing termination condition is to use an empirically determined, fixed number of transmitters.

7.2.8 Sampling rate mismatch

Sampling rate mismatch refers to the phenomenon that, across different devices, the sampling rates of the audio hardware are not exactly the same. In this section, first, we show that real devices have non-identical sampling rates and estimate the extent of the mismatch. Then, we run simulations to explore how the sampling rate mismatch affects the positioning accuracy.

Method to estimate the sampling rate mismatch

Let f_A be the sampling rate of a device A. Then, we can define the sampling rate mismatch between two devices A and B as

$$\epsilon_{AB} := \frac{f_B}{f_A} - 1 \tag{7.2}$$

Using two "clock reference" signals transmitted by a given device (called clock reference device, CRD), the sampling rate mismatch can be computed for any pair of receivers (R_1, R_2) based on Ono et al.'s work in [63] as follows. After signal analysis, both R_1 and R_2 count the number of audio samples between the two signals of CRD. Let this sample count be Δ_{R_1} and Δ_{R_2} on R_1 and R_2 respectively. Then, the sampling rate mismatch can be estimated as follows:

$$\hat{\epsilon}_{R_1R_2} = \frac{\Delta_{R_2}}{\Delta_{R_1}} - 1 \tag{7.3}$$

Accuracy of the estimation of ϵ

Consider the scenario where R_1 and R_2 receive a signal pair from a CRD, and suppose that $\epsilon_{R_1R_2} = 0$. Assuming perfect signal analysis, $\epsilon_{R_1R_2} = 0$ implies that $\Delta_{R_1} = \Delta_{R_2}$. Let x denote this common value. Now suppose that the signals' arrival timestamps can be estimated within ±5 samples. Due to this inaccuracy in signal analysis, Δ_{R_1} and Δ_{R_2} will likely be different, affecting the calculation of $\hat{\epsilon}_{R_1R_2}$. In the worst case, $\Delta_{R_1} = x + 10$ samples and $\Delta_{R_2} = x - 10$ samples,

which leads to an estimated sampling rate mismatch of

$$\hat{\epsilon}_{R_1R_2}(x) = \frac{x - 10}{x + 10} - 1 \tag{7.4}$$

Observe that the estimated sampling rate mismatch depends on the (correct) time gap x. As the time gap approaches zero, the estimated sampling rate mismatch becomes extremely inaccurate: it will approach -2, which would imply that $f_{R_2} \approx -f_{R_1}$, i.e., one of the sampling rates is negative! Additionally, as the time gap approaches positive infinity, the estimated sampling rate mismatch approaches the correct value 0.

Overall, we can conclude that (1) the accuracy with which we can estimate ϵ depends on the accuracy of signal analysis, and (2) as we increase the time gap between the clock reference signals, the impact of signal analysis inaccuracy becomes smaller. Thus, for more accurate sampling rate mismatch estimation, it is better to have a larger gap between the clock reference signals.

Experimental evaluation of the sampling rate mismatch

Consider two clock reference signals from a CRD and two receiver devices R_1 and R_2 whose sampling rate mismatch is not zero. The two devices will observe a different number of audio samples (Δ_{R_1} and Δ_{R_2}) between the CRD's two audio signals. Let δ_{R_1,R_2} denote this difference:

$$\delta_{R_1,R_2} := \Delta_{R_2} - \Delta_{R_1} \tag{7.5}$$

If we increase the time gap between the CRD's two signals, δ_{R_1,R_2} should also increase.

In order to confirm that real devices exhibit a non-zero sampling rate mismatch, we need to detect this increase in δ experimentally. Towards this end, we prepared 5 Motorola Nexus 6 phones [27], 5 Sony G8441 Xperia XZ1 Compact phones [31] and 5 RPis [23] in an office. In order to minimize the impact of signal processing issues, the office was quiet, all pairwise distances were at most a few meters, and the devices had a clear line of sight to each other. Then, we conducted experiments with increasing time gaps between signals: one without any artificial delay, and another 3 experiments with an additional delay of 10, 20 and 30 seconds after each signal transmission. In each experiment, first, each phone transmitted one signal, and then each phone transmitted another one in the same order (two audio signals per phone in total). Since we did not equip our RPis with speakers, they did not transmit signals. All transmissions were recorded by all devices.

Using the method described above, we estimate δ and ϵ for all possible combinations of CRD candidates and microphone pairs — more formally, for all elements of the following set:

$$\{(CRD, R_1, R_2) \mid CRD \in Transmitters \wedge R_1, R_2 \in Microphones\} \tag{7.6}$$

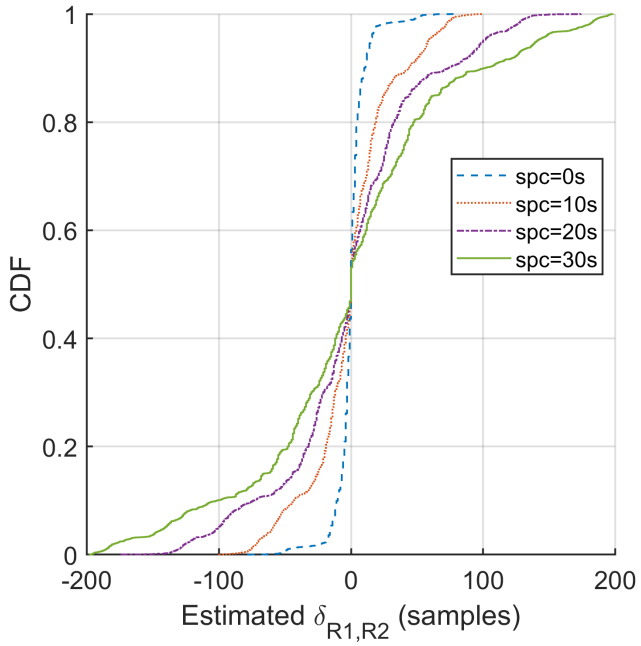

Figure 7.11: Distribution of δ as a function of the time gap between signal transmissions (spc)

The distribution of δ is shown on Figure 7.11. We can see that the value of δ increases approximately linearly as we increase the time gap between the signals, which confirms that the sampling rate mismatch is not zero. In fact, the figure suggests that, in some cases, δ can be over 100 samples. As a reference, at 48 kHz sampling rate, sound travels over 70 cm during 100 audio samples, suggesting that the phenomenon deserves attention.

Now that we have confirmed that our devices have a non-zero sampling rate mismatch, we estimate its extent by computing the ϵ values. Since ϵ is just a (shifted) ratio of two sampling rates, it does not depend on the time gap between the two clock reference signals. However, due to the dependence between the time gap and the signal analysis inaccuracy, the *estimated* ϵ does depend on the time gap. More specifically, as we increase the time gap, the distribution should converge to the correct distribution. Figure 7.12 shows the empirical distribution of ϵ for several time gap values, which confirms our above reasoning. Since the distribution does not change much if we increase the time gap from 20 s to 30 s, we conclude that the 30 s time gap is sufficient for estimating ϵ.

Dependence of the sampling rate mismatch on the hardware We also evaluated how the sampling rate mismatch depends on the type of device used. In addition to comparing different device types, we also compared different devices of the same type. For the single experiment with the 30-second additional delay per signal (which we consider the most accurate one for reasons described above), Figure 7.13 shows the distribution of the estimated sampling rate mismatch for all combinations of device types.

Based on the curves that consider only devices of the same type, we can see that the Xperia

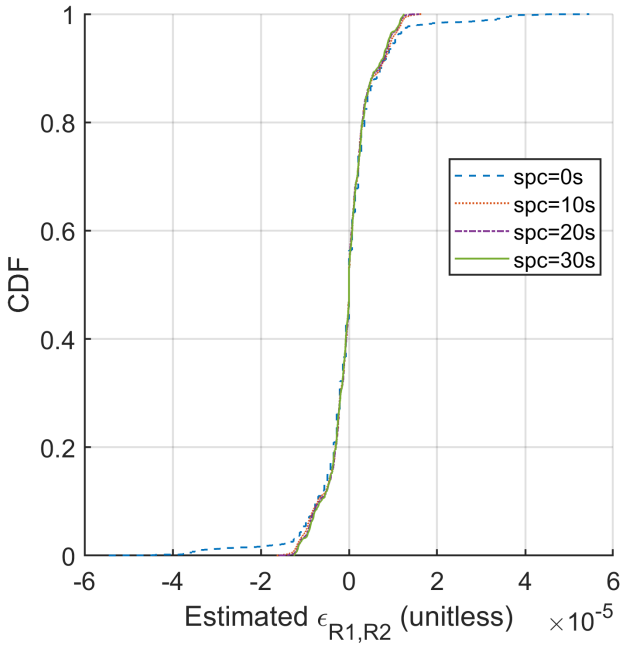

Figure 7.12: Distribution of ϵ as a function of the time gap between signal transmissions (spc)

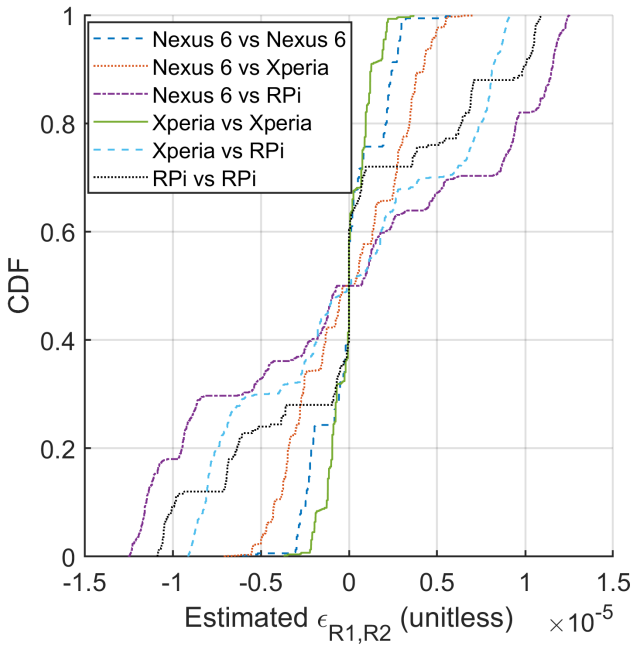

Figure 7.13: Distribution of ϵ as a function of the device type. "Nexus 6" refers to Motorola Nexus 6 smartphones. "Xperia" refers to Sony G8441 Xperia XZ1 Compact smartphones. "RPi" refers to Raspberry Pi devices, the sampling rates of which are controlled by the attached Sabrent AU-MMSA USB audio adapter. The data is from the experiment with the 30-second additional delay per signal.

phones have the most stable sampling rate, followed by the Nexus 6 phones, and that the RPis have much less stable sampling rate than both phone models. Additionally, device pairs involving RPis tend to have a significantly higher mismatch than device pairs involving only smartphones. This result confirms that our RPis have lower-quality audio hardware than our phones, which explains at least part of the discrepancy between our simulations and real experiments.

Another observation we can make is that the Nexus 6 vs. RPi group contains higher values than both the Nexus 6 group and the RPi group individually. Similarly, the Xperia vs. Nexus 6 group contains higher values than both the Xperia group and the Nexus 6 group individually. One possible explanation is that the sampling rates of different device types are clustered at slightly different values.

Sensitivity simulations (sampling rate mismatch and signal spacing)

Given a pair of receivers, the extent of the sampling rate mismatch determines how fast the two receivers' audio sample clocks diverge per unit of time. The total duration of a localization protocol determines how long this divergence goes on. Thus, the higher the sampling rate mismatch and the longer the time gap between the first and last signal of a localization protocol is, the more noise will be added to the perceived sample counts of the two receivers, negatively affecting the estimated distances and distance differences, and, consequently, the positioning accuracy as well.

In this section, we evaluate Sonoloc's sensitivity to these two factors using simulations. We define the set of the sampling rate mismatch values as

$$x = \{0, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\} \tag{7.7}$$

Then, for each simulated device A, the sampling rate is calculated as

$$f_A = 48\,\text{kHz}(1 + UniformRandom[-x, x]) \tag{7.8}$$

We define the set of signal spacing values as

$$x = \{10^{-4}\,\text{s}, 1\,\text{s}, 5\,\text{s}, 10\,\text{s}, 60\,\text{s}, 600\,\text{s}, 86\,400\,\text{s}\} \tag{7.9}$$

and use them directly as the simulated time gap after each signal (more precisely, the gap between the start times of two neighboring transmissions). We chose the range of values in such a way that they include the "realistic" value range. In our simulations, we assume that the sampling rate mismatch is linear, i.e., each device has a slightly different, but fixed, sampling rate.

For each combination of the above two sets of values, we ran 130 simulated experiments. In these experiments, we chose 15 transmitters iteratively and otherwise used our default settings. Then, we computed the max IPDE across all experiments and all devices of the given combination. These simulations resulted in a matrix of max IPDEs, shown on Figure 7.14.

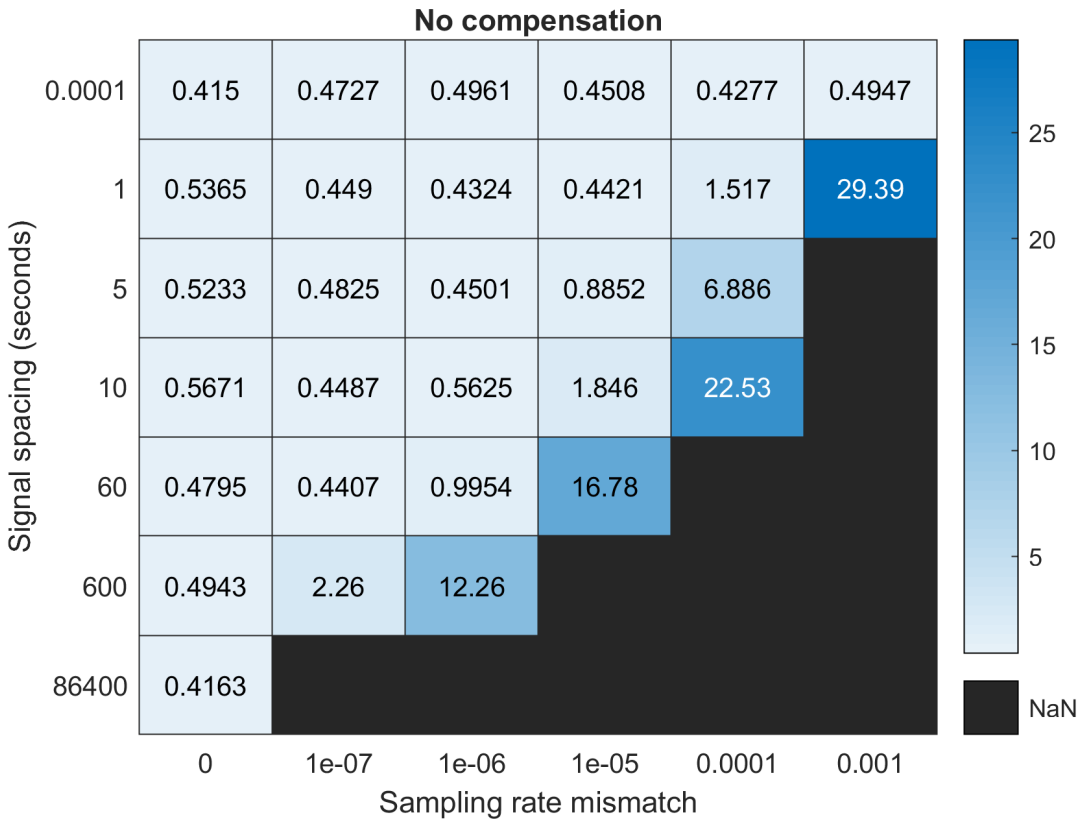

Figure 7.14: Max. IPDE in meters from simulations with varying values of sampling rate mismatch and signal spacing. The algorithm does not attempt to counter the effects of the sampling rate mismatch.

When at least one experiment of a given configuration failed (could not localize all devices), the corresponding cell contains NaN.

As can be seen clearly from the results, the accuracy is good as long as the overall effect of the sampling rate mismatch is small. However, as the effects of the sampling rate mismatch become significant, the max IPDEs also increase. As we move towards the bottom right corner of the plot (high sampling rate mismatch over a long time), experiments start to fail.

In these sensitivity experiments, due to the way we determine the sampling rate of each device, a sampling rate mismatch of x will lead to an approximately twice as high *maximum pairwise mismatch* (the metric shown on Figure 7.13). On the smartphones we tested, the highest observed pairwise mismatch was around 7×10^{-6}, which corresponds to a sampling rate mismatch of about 3.5×10^{-6} on the X-axis of Figure 7.14. (If we include RPis as well, these numbers would be 1.25×10^{-5} and 0.625×10^{-6}, respectively.) Regarding the time gap between transmissions, we believe that 5 seconds or less is realistic (also accounting for the need for certain computations in between transmissions). If we observe the matrix in Figure 7.14, we can see that these values are small enough to maintain good accuracy. However, even a relatively small increase in these factors (e.g., due to lower-quality hardware such as RPis) runs the risk of increasing the max IPDE significantly.

7.2.9 Sensitivity to Sonoloc's configuration parameters

Sonoloc has a number of configuration parameters, the values of which we determined empirically. In the following, we report results on how these parameters affect Sonoloc's positioning accuracy. These results were obtained from both simulated and live experiments.

Transmitter selection algorithm: As discussed in Chapter 3, a good choice of transmitters is important to achieve high accuracy. We compared different transmitter selection algorithms on many device layouts of the types described above. To get a sense of the best and worst possible algorithm, we also tried a large number of random transmitter selections on a given device layout and recorded the best and worst of N random selections, for $N = 100$. In terms of maximal IPDE, Sonoloc's transmitter selection algorithm achieved 0.57 m, while the best of 100 achieved 0.35 m. The worst of N selections has a maximal IPDE of almost 11 m, underscoring the importance of transmitter selection. In addition, a simple random selection of transmitters resulted in a maximal IPDE of 8.4 m, which shows that intelligent transmitter selection is important.

Microphone selection: Since different signals cannot always be detected by the same microphone, insisting on a given microphone for all signals gives much worse accuracy than choosing dynamically. Data from our real experiments confirms this intuition. (In real experiments, a microphone's signal quality depends on the position and orientation of the device, which is outside the scope of our simulations. Thus, we only evaluated the microphone selection using real experiments.)

TOA-based iterative refinement: We tried allowing a maximum of 0, 10, 100, 1000, and 3000 iterations with otherwise default parameters. In our real experiments, performing at least 10 iterations improves the maximum IPDE by around 5 cm compared to not using the iterative refinement, but further iterations have negligible impact on IPDEs. In simulations, the number of iterations did not have a statistically significant effect on IPDEs. Based on these results, we concluded that 10 iterations are sufficient.

TDOA-based iterative refinement: We tried allowing a maximum of 0, 10, 100, 1000, 2000, 5000, 10000, and 20000 iterations with otherwise default parameters. We observed that performing the iterative refinement improves the positioning accuracy, but, beyond 2000 iterations, the effect becomes negligible. Thus, Sonoloc caps the number of iterations at 2000. Additionally, to save on runtime, we stop the iteration when the change in the objective function becomes so low that floating-point precision issues get in the way of further improvements. With this additional condition in place, at our current default setting, the actual number of iterations performed is typically less than 1000, but more noise in the input data tends to increase this number.

Eliminating invalid distance differences: We ran experiments with and without filtering the estimated distance differences based on triangle inequality. In the case with the filtering enabled, we tried the following tolerance factors: 1, 1.2, 1.5, 2, 3, 10, and 100. In simulations, this filtering had negligible impact on the localization accuracy. In real experiments, some estimated distance differences have very high error, thus, without the filtering and with factor 100 (where the filtering only triggers in extreme cases), the IPDE increases to over 65 m. Overall, tolerance factors between 1 and 10 gave similar accuracy. In our experiments (with default parameters), the filtering eliminated ca. 1% of all distance differences within a given experiment.

Number of initial transmitters: To explore the impact of the size of the initial, randomly chosen set of transmitters on Sonoloc's positioning accuracy, we ran experiments with the following numbers of initial transmitters: 4, 6, 7, 8, 9, 10, 13, and 15. According to our simulation results, when the number of initial transmitters is between 4 and 10, we get similar accuracy (the max. IPDE fluctuates around 45 cm–55 cm). With 13 or more initial transmitters, the number of strategically chosen transmitters decreases to 2 or less, and the randomly chosen transmitters dominate the transmitter set. Indeed, the max. IPDE increases above 1m. (Trace-based evaluation does not make sense because it ends up with the same 15 transmitters anyway, giving the same result.) Based on these results, Sonoloc uses 6 initial transmitters.

7.2.10 Sonoloc runtime overhead

In this section, we evaluate Sonoloc's runtime overhead. First, we consider the processing overhead of the signal analysis phase. Then, we evaluate the processing overhead of the localization algorithm. Finally, we discuss the limits of reducing the end-to-end delay of the protocol.

Computation time of the signal analysis

We define the computation time of the signal analysis as the combined execution time of the correlation and the peak detection stages. We measured the runtime of the signal detector on 242 recorded audio waveforms during one of the live experiments (lecture hall, background music). Since the signal analysis overhead depends largely on the length of the recorded audio waveform, we normalized the runtime to the length of the recordings.

On a Dell Precision T1700 computer with 16 GB of RAM and an Intel(R) Xeon(R) CPU E3-1246 v3 @ 3.5 GHz, the signal detector took a minimum, median, maximum of 0.16, 0.19, and 0.3 seconds, respectively, per second of recorded audio waveform.

Computation time of the localization algorithm

First, we provide some information on the complexity of the localization algorithms involved.

Closed-form method for transmitter localization: The most computationally intensive part is the singular value decomposition (SVD) [35] of the matrix of pairwise distances. The complexity of computing the SVD of an $m \times n$ matrix is $O(min(m \cdot n^2, n \cdot m^2))$. Given N transmitters, the matrix of pairwise distances is of size $N \times N$, so the computational complexity of the closed-form method is approximately $O(min(N \cdot N^2, N \cdot N^2)) = O(N^3)$.

Iterative refinement for transmitter localization: Since the number of iterations we execute is very small, its impact on the runtime is negligible.

On Figures 7.15 and 7.16, the bottom, center and top of each box indicate the 25th, 50th, 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers, and each outlier is plotted using a circle. A point is an outlier if its value is greater than $q_3 + 1.5(q_3 - q_1)$ or less than $q_1 - 1.5(q_3 - q_1)$ where q_1 and q_3 are the 25th and 75th percentiles of the sample data, respectively.

Closed-form method for passive device localization: The most computationally intensive parts are the matrix pseudoinverse and vector-matrix multiplication. Figure 7.15 shows the results of our empirical runtime measurements. These results suggest that the computation time is approximately a linear function of the number of transmitters. Although Sonoloc would not

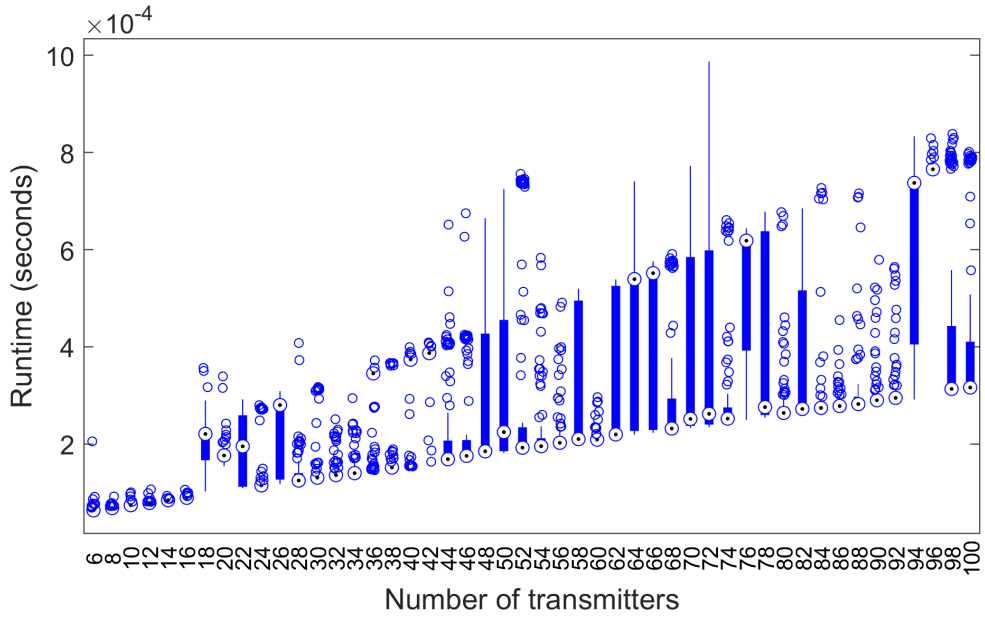

Figure 7.15: Box plot of the runtime of localizing 1 passive device using the closed-form method without iterative refinement. For each number of transmitters on the X-axis, we ran 100 simulated experiments. (Measured on the same Dell Precision T1700 computer as in the previous section.)

# Devices	Average	Std. Dev.	Max.
20	3.2	0.5	3.8
100	10.4	5.7	22.9
500	48.5	13.4	68.6

Table 7.2: Localization runtimes in seconds.

need more than 15 transmitters, the plot's X-axis goes up to 100 transmitters in order to highlight the linearity of the computational complexity (which would otherwise be lost in the noise).

Iterative refinement for passive device localization: The computation time depends on the number of iterations actually executed, which is very input-dependent. To get an estimate, we performed the following simulation. We obtained random device positions for 15 transmitters and 1 passive device in a 20 m × 20 m space. We added random noise to the distance differences (standard normally distributed noise multiplied by 0.25 m). Then, we estimated the position of the passive device via the closed-form method, and used this position as the starting point for the iterative refinement. Then, we started a timer, ran the iterative refinement and took timestamps every 50 iterations. Then, we repeated the entire process 200 times. Figure 7.16 shows the distribution of the measured runtimes. We can see that the runtime increases approximately linearly as a function of the number of iterations. Additionally, some experiments converge after a few hundred iterations; for these, the runtime stops increasing.

The above reasoning implies that the total runtime of the localization algorithm depends on

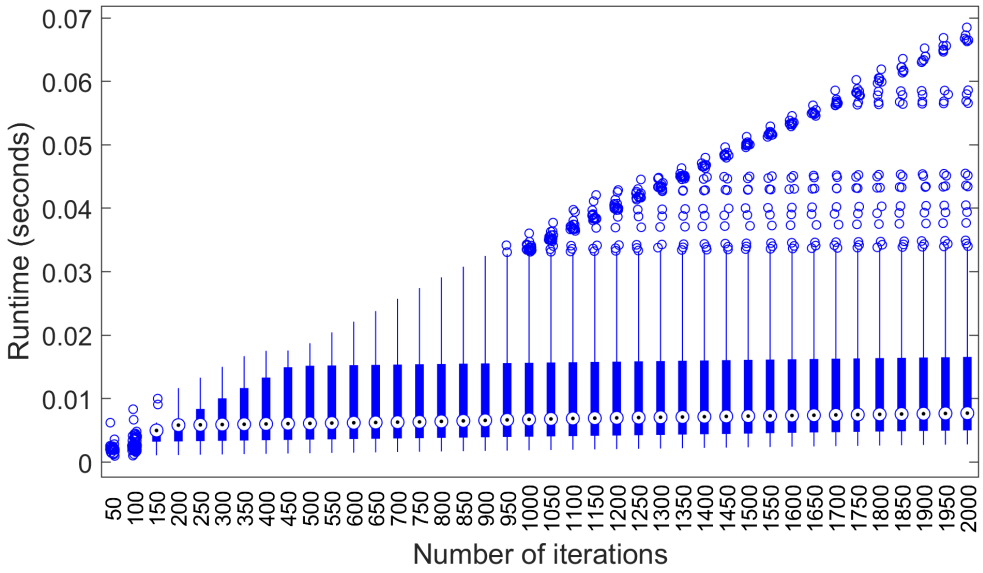

Figure 7.16: Box plot of the runtime of the iterative refinement method as a function of the number of iterations. 200 simulated experiments. (Measured on the same Dell Precision T1700 computer as in the previous section.)

the number of transmitters and the number of passive devices. We generated 10 random device layouts with 20, 100, and 500 devices in a 20 m × 20 m square space, and ran the algorithm on each layout with 15 transmitters. In Table 7.2, we report average and maximal runtime, as well as the standard deviation across 10 sequential runs of the algorithm.

The focus in our prototype implementation so far has been on accuracy and robustness. There are numerous, unexplored opportunities for optimizing the localization runtime. In addition, both the transmitter localization and the passive device localization can be parallelized. For instance, the localization of different passive devices is independent and can be trivially executed in parallel.

End-to-end protocol delay

The lowest achievable end-to-end protocol delay is limited by the following requirements:

- Receivers must be able to distinguish and identify incoming transmissions correctly. Since, in Sonoloc, a single reference signal is transmitted multiple times using time-division multiplexing, for reliable detection, the time overlap among different audio signal transmissions (including their echoes) must be minimized. Alternative, more time-efficient forms of multiplexing are possible (see Section 8.3), but they are beyond the scope of this thesis.
- After the initial set of transmissions, time between subsequent transmissions must allow for signal analysis (including signal upload), localization, and transmitter selection based on the output of the localization.

Since Sonoloc was designed for static scenarios, minimizing this delay has not been a focus of our implementation.

Chapter 8

Discussion and additional challenges

In this chapter, we discuss some additional challenges raised by Sonoloc, and propose some initial ideas about how they might be tackled in the future. First, we discuss Sonoloc's privacy implications. Then, we discuss some issues related to Sonoloc's geometric constraints. We continue by discussing how to improve Sonoloc's signal analysis techniques, and how to add support for incremental positioning. Additionally, we discuss some alternative ways to implement Sonoloc's functionality, and discuss some usability challenges that can benefit from further study. The hypothetical systems we propose in this section do not exist at the time of writing. We leave the implementation and evaluation of these ideas to future investigations.

8.1 Privacy considerations

To facilitate device discovery, users need to reveal their Bluetooth MAC address to nearby devices. Since Bluetooth Low Energy supports temporary MAC addresses (see Part B: Link layer specification, Section 1.3: Device Address in [78]), this issue can be mitigated by simply choosing a new MAC address after the position map is computed.

In our implementation, users also need to reveal their network/IP address to the localization server. If this is a concern, users should hide their network address using existing, orthogonal techniques such as Tor [81, 70] or a VPN service [62, 30, 36]. Alternatively, a decentralized implementation (discussed in Section 8.5) is possible at the cost of increased resource consumption.

In summary, by providing an ephemeral, context-specific network address and an optional nickname along with their relative position, participating devices can communicate within a given context without exchanging any long-term identifiers or personally identifying information at all.

8.2 Geometric constraints

In this section, we discuss issues related to Sonoloc's geometric constraints. First, we talk about the orientation ambiguity of the position map. Then, we give a brief outlook on the possibilities of 3D positioning. Finally, we discuss problematic device layouts.

In order for a relative position map to be useful to users, it is important that the orientation of the map match users' mental image of the scene. However, any 2D relative position map has an inherent orientation ambiguity that raises the following two issues. The first one is the *flipping problem*: both a map corresponding to the *top-down* view of the room and a map corresponding to the *bottom-up* view of the room are consistent with the observed distances and distance differences. The second one is the *rotation problem*: distance information does not provide any hints about how the position map should be rotated to be meaningful to the user. For both issues, the challenge is to find the most intuitive and simple way that minimizes the amount of work done by the user.

If a user's device happens to be on a symmetry line of the position map, then the task of identifying participants becomes even more difficult. For instance, consider a lecture hall where people are sitting in a square-shaped grid. Then, Alice, who sits in the middle, looks at the position map, and she wants to contact Bob, who is sitting in the corner of the lecture hall. Without additional information about the orientation of the displayed map, it is difficult for Alice to find out which of the four corners on the map corresponds to Bob.

In the following, we present some initial ideas on how to deal with these problems using visual cues and rotation sensor data.

8.2.1 Flipping problem

For a given map, the flipping problem needs to be solved only once. Once a user has resolved it, the server can provide the correct version to all participants upon request. We suggest that the initiator IN performs this task. Since the map is either flipped one way or the other, the output of the process is binary. Since users are used to looking at maps from above (think geographic maps), we propose to determine the orientation of the relative position map that is consistent with a top-down view of the room.

The key idea to solving the flipping problem is to choose 2 landmark devices L1 and L2, observe their positions relative to IN both in the real room and on the position map, and compare the two observations. We will perform this comparison based on the angle between the $IN \rightarrow L1$ and $IN \rightarrow L2$ vectors. We define this angle more formally as follows:

Definition 1. *Let C,P,Q denote device positions in 2D space. Initialize a vector to the $C \rightarrow P$ vector, and rotate it around C in clockwise direction until it reaches the $C \rightarrow Q$ vector. We define $rot_{CW}(C, P \rightarrow Q)$ to be the angle of this rotation (see Figure 8.1).*

In the following, we present two concrete approaches to solving the flipping problem.

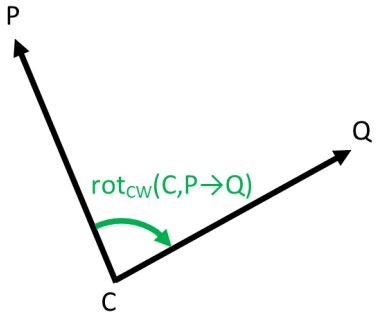

Figure 8.1: Definition of $rot_{CW}(C, P \rightarrow Q)$

Approach 1

The first approach augments localization with some basic visual cues. In this approach, the initiator IN chooses 2 landmark devices L1 and L2, and turns towards them so that, the angle α defined by (L1, IN, L2) in this order is smaller than 180°. Then the landmark devices vibrate, show the numbers "1" and "2" on their screens with a big font, and their users hold their devices up high so that IN can see them. The system asks IN whether L2 is on the left side or right side of L1, and IN enters the answer.

After performing the above sequence, it becomes possible to resolve the flipping problem as follows. We derive a constraint for $rot_{CW}(IN, L1 \rightarrow L2)$ based on IN's response to the "left or right" question.

- If the user says "L2 is on the right", then $rot_{CW}(IN, L1 \rightarrow L2) < 180°$.
- If the user says "L2 is on the left", then $rot_{CW}(IN, L1 \rightarrow L2) > 180°$.

Flipping the map "inverts" all clockwise rotation angles in the sense that, if $rot_{CW}(IN, L1 \rightarrow L2) = \alpha$, then, on the flipped map, it will be $\alpha' = 360° - \alpha$. Consequently, if $\alpha < 180°$, then, on the flipped map, $\alpha' > 180°$, and vice versa. Thus, in order to determine the correct map variant, we need to compute $rot_{CW}(IN, L1 \rightarrow L2)$ for both map variants, and choose the one that matches the user's response.

Although this approach works, it is limited to angles smaller than 180°. The user's statement "L2 is on the right" is a user-friendly way of expressing the more precise statement $rot_{CW}(IN, L1 \rightarrow L2) < 180°$. As illustrated on Figure 8.2, if we allow angles greater than 180°, then it could happen that L2 "is still on the right", but $rot_{CW}(IN, L1 \rightarrow L2) > 180°$, which would confuse our method. If we use a rotation sensor, we can determine the rotation explicitly, without relying on potentially ambiguous statements by the user. We outline this idea in approach 2 below.

Approach 2 (using rotation sensor)

Similarly to approach 1, the initiator IN chooses 2 landmark devices L1 and L2, and their owners hold the devices up high. However, now angles larger than 180° are also acceptable, making this

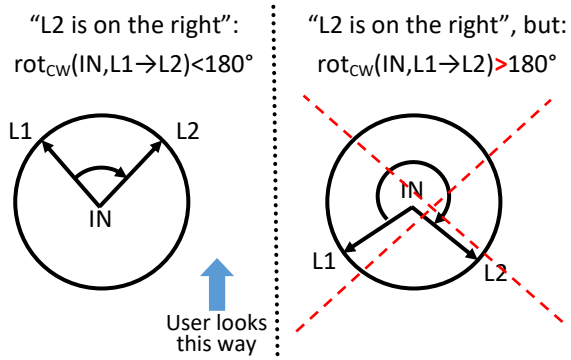

Figure 8.2: Limitation of approach 1. The user is looking towards the top of the figure.

approach more general than approach 1. IN makes the device screen face upwards. IN points at L1 and starts recording the device's orientation. While keeping the device facing upwards (which is necessary to track the rotation accurately), IN rotates the device to point towards L2 and stops recording the device's orientation.

In order to determine the correct flipping, we rely on the rotation sensor, which can observe the direction and angle of rotation that was required to turn from L1 to L2. Without loss of generality, assume that this rotation is a clockwise rotation.

Assume that the rotation sensor determined this angle to be $rot_{CW}(IN, L1 \rightarrow L2) = \alpha$. On a correctly flipped map, this angle will be approximately the same. However, on an incorrectly flipped map, the angle will be $\alpha' = 360° - \alpha$. If we initialize a vector to equal the $IN \rightarrow L1$ vector on the map, and perform the rotation α we observed in reality, we can determine the correct flipping by checking whether it also arrives approximately at the $IN \rightarrow L2$.

This method assumes that α and $360° - \alpha$ are distinguishable. However, these two angles cannot be distinguished in the following cases:

1. $\alpha = 0°$: Since a 0° rotation is the same as a 360° rotation, $\alpha = 360° - \alpha = 0°$.
2. $\alpha = 180°$: In this case, $\alpha = 360° - \alpha = 180°$.

Additionally, due to the limited accuracy of (1) the localization, (2) the rotation sensor and (3) users' ability to rotate the phone at the required angle, if $\alpha \approx 0°$ or $\alpha \approx 180°$, then α and $360° - \alpha$ are difficult to distinguish.

8.2.2 Rotation problem

Now let us assume that the flipping problem is resolved. After running Sonoloc, every participant has a position map and a mental image of the room. However, the two are not necessarily rotationally aligned. During or after the event, users may want to rotate the map and align it with their mental image of the scene. Additionally, since each user may remember the scene differently, users should have the ability to rotate the map according to their own preferences independently from other participants.

The challenge is how to provide useful cues for a user U to be able to perform this rotation even after the event. In the following, we propose two possible approaches to address this challenge.

Approach 1: record the $U \rightarrow IN$ vector The first approach is applicable if U remembers where IN was located during the event. Since the system knows the location of both U and IN on the position map, it can record the $U \rightarrow IN$ vector on the map. U can later look at the map, remember the position of the vector inside the room, and rotate the map to align it with his/her memory. For instance, U may think, "when I was looking towards the whiteboard, IN was on my left side, so, if I want the whiteboard to be on the top, I need to rotate the map so that IN is on the left side".

Approach 2: take a photo in the direction of $IN \rightarrow L1$ The second approach uses additional visual cues in order to eliminate the need to remember the location of IN. In this approach, IN takes a photo in the direction of L1. The system knows the location of both IN and L1, so it can record the $IN \rightarrow L1$ vector on the position map. Since the photo was also taken in the direction of the $IN \rightarrow L1$ vector, the photo serves as a visual cue to help U associate a given vector on the position map with a visual view of the room.

We could also allow IN to take multiple photos in different directions. First, IN takes a photo in the direction of L1. Then, IN starts turning the camera left or right, and takes photos while turning until everyone in the room is covered. Using the rotation sensor, the app can track the device's rotation relative to the $IN \rightarrow L1$ vector and associate each photo with an arrow on the map.

A potential user interface for the multi-photo approach would be the following. The top half of the screen shows a photo of the room. The bottom half shows the relative position map and an arrow starting from IN. The arrow indicates the direction in which the currently displayed photo was taken. In order to move to the next photo angle, the user makes a horizontal swipe gesture. If IN takes a panorama photo, the photo angle could be adjusted smoothly.

8.2.3 Relative positioning in 3D space

Sonoloc currently works in 2D space. In some situations such as open spaces that span multiple floors, it would be useful to be able to produce 3D coordinates. As we briefly mentioned in Section 2.2.1, closed-form solutions exist, but they also have a number of disadvantages compared to 2D variants: (1) they need a higher number of transmitters, (2) their computation time is higher, and (3) the transmitters need to be spatially distributed along all three axes of the 3D space. Given these disadvantages, it is not clear whether a practical system can be built. The flipping and rotation problems also persist in 3D space and, due to the additional dimension, may be more difficult to solve. Finally, there is the challenge of communicating a 3D relative

position map to the user in an intuitive way.

8.2.4 Identifying problematic device layouts

We know that there are some rare corner cases where the presented method of localization cannot work. For example, if all transmitters are located along a straight line, then the localization of a passive device based on distance differences is not possible because the constraints we derive are simply insufficient to pinpoint a passive device's location. Identifying additional classes of problematic device layouts would enable us to understand the limits of our localization method better.

8.3 More sophisticated signal analysis techniques

Simultaneous transmission of audio signals: Sonoloc currently transmits the audio signals sequentially via time-division multiplexing. In order to improve the protocol's end-to-end delay, one could transmit the initial set of audio signals simultaneously using a more time-efficient form of multiplexing such as frequency division or code division multiplexing. ([58] presents a comparison of these multiplexing techniques in the context of indoor localization.) However, in this case, multiple signals would share the same spectrum, decreasing the SNR. In the case of frequency division multiplexing, the bandwidth available to each individual signal would decrease, which would go against the "wideband" requirement discussed in Section 5.2. Since the bandwidth of the audio equipment on commodity smartphones is limited, it is unclear whether such an approach would be practical. Additionally, simultaneous transmission is only feasible for the initial set of signals because, in order to choose the remaining transmitters, Sonoloc relies on data from the previous transmissions.

Using multiple speakers per device: Sonoloc currently transmits on only one speaker of each transmitter. Due to obstructions, a speaker may not have a direct line of sight to either microphone of a receiver, which would affect the signal analysis. The robustness of signal analysis could be improved by emitting an acoustic signal on multiple speakers of a transmitter at the same time. Assume that devices have 2 microphones. When using only one speaker, we have 2 direct paths from transmitter to receiver: one from local speaker 1 to remote microphone 1, and one from local speaker 1 to remote microphone 2. If at least one of these 2 paths is unobstructed, the signal traveling on that path can be detected at the receiver. If we use 2 speakers, the number of paths doubles. Thus, successful signal detection requires only one of 4 paths to be unobstructed. However, there is a tradeoff: in order to be able to extract individual signals, some form of multiplexing (frequency division or code division) is required. Again, due to multiple signals sharing the information capacity of the same acoustic channel, both signals would experience a decrease in SNR.

8.4 Incremental positioning

Sonoloc could be extended to support incremental positioning, i.e., to be able to position a new device that joins an existing, already positioned set of devices. A possible solution would be as follows. The new device stops moving, and then transmits an audio signal. By having transmitted signals for the previous position map, the existing transmitters have effectively synchronized their *recording* channel [63], so we can obtain time difference of arrival (TDOA) information for the existing transmitters and the new device. The existing transmitters will act as a *microphone* array and the new device will act as a source. Essentially, the problem becomes an instance of "source localization with known microphone positions", which is a classical problem [80]. Thus, we can apply an existing TDOA-based localization algorithm to determine the position of the new device.

8.5 Alternative implementations

In this section, we first discuss how one can decentralize Sonoloc and remove the need for a centralized server. Then, we propose a vision-based alternative to audio signals that, with some tradeoffs, could support the use cases we envisioned for Sonoloc.

8.5.1 Decentralized implementation

While the current Sonoloc prototype delegates all heavyweight computations to a server, it is also feasible to run some or all of the computation on the devices themselves. The heavyweight computations are the signal analysis and the localization. For each of these, we can decide to run it on the devices instead of running it on the server.

Regarding the signal analysis, it is possible to implement it on the mobile devices. Then, instead of uploading large wave files to the server (which uses a large amount of network traffic), the signals' time delays could be calculated on the mobile devices, and then only those would need to be sent to the server for further processing. However, due to the computation cost of the signal analysis, it is unclear which option is better in terms of computation time and energy usage.

Regarding the localization algorithms, they can also be implemented on the devices as follows. First, the initiator obtains the output of the signal analysis phase, either from the server or from the other devices. Then, the initiator's device runs the transmitter localization. Then, the initiator provides each passive device with the inputs required to localize that passive device. Since each passive device localizes itself, resource consumption is distributed across the devices. Finally, passive devices send their positions back to the initiator, who then assembles and broadcasts the position map. While such a decentralized approach is feasible, since the inputs and outputs are small, it is cheap to offload the localization computations to a cloud

service, and a cloud-based implementation would have much lower energy cost on the local devices.

In order to arrive at a fully decentralized approach, in addition to doing the above computations on the devices, one could also assign the role of coordinating signal transmissions, which is currently assumed by the server, to one of the devices (e.g., to the initiator). However, we believe that this additional degree of decentralization would not improve Sonoloc's deployability in any way. Network access and cloud services are widely available, and they do not count as *local, on-the-premises* infrastructure. Therefore, they are not an impediment to deployment.

8.5.2 Vision-based approach

Instead of using audio signals, one could build a system based on face recognition technology for identifying and contacting the people photographed. Such a system could be implemented by modifying I-Pic [1], which is a system that blurs faces in photos based on the photographed user's privacy preferences.

A simple solution without privacy would be the following. Each device would broadcast the owner's facial signature (e.g., a feature vector) like in I-Pic, along with some sort of contact information. The initiator takes one or more photos. In order to help a user choose from the faces on the photo, the faces would not be blurred. By comparing the received facial signatures to the faces detected in the photo, the system would be able to maintain a mapping between a face in the photo and its contact information. Then, when the user selects a face, the system would know how to contact the corresponding person.

The privacy properties of the solution could be strengthened as follows. Like in I-Pic, faces on photos would be blurred. Devices would broadcast a temporary address instead of their permanent contact information. Such a temporary address could be negotiated using an anonymous key exchange system such as SDDR [48]. The drawback of the privacy-enabled approach is that the user would need to select from blurred faces in a photo. It would still be feasible to select groups of people based on where they are in the photo, but choosing specific people would become more difficult.

Just as the audio-based approach depends on the correct reception of audio signals, for this visual approach to work, the face of each person of interest must be clearly captured in at least one of the photos taken by the initiator.

8.6 Better usability

The current version of Sonoloc is a prototype built for evaluating the feasibility of the ideas we propose in this thesis. Thus, there are some additional usability challenges that can benefit from further study. In this section, we mention some of these and present some initial ideas for resolving them.

Choosing the correct set of devices to participate in the protocol: In some settings, if everybody within Bluetooth range participates in the localization protocol, some irrelevant devices may accidentally be included. Since walls do not completely block electromagnetic waves, Bluetooth might detect devices in the neighboring room, one floor above or one floor below. If those devices are chosen as transmitters, they will unexpectedly transmit an audio signal. A possible solution is to choose the initiator's device as the first transmitter. Then, the initiator contacts every device in Bluetooth range, and asks them whether they "heard" the signal, i.e., whether the signal detector triggered on anything during the corresponding time interval. Then, the protocol will proceed with only the devices that detected the initiator's first signal, and other devices will be excluded.

Improving hardware compatibility: Up until now, we have only experimented with Motorola Nexus 6 and Sony G8441 Xperia XZ1 Compact smartphones, but there are many other, commonly used devices on the market with varying hardware and software characteristics. These differences might include the following:

- Transmission power of the speaker
- Signal to noise ratio of the microphone
- Frequency response curve of speaker and microphone
- Number of microphones
- Position of speaker and microphone, including speaker-microphone distance
- Stability of the audio sampling rate

By trying out other smartphone models from different manufacturers and characterizing the observed differences, one could adjust Sonoloc to work with a wider range of devices.

User interface: In order to be useful to users, Sonoloc needs to visualize the position map intuitively. Our initial idea is to represent each device by a small circle. The app would display both the device itself and the initiator with a unique color or shape. The user would be able to pan, zoom and rotate the map. Clicking on a node would toggle the selection of the corresponding device. The user would be able to select multiple devices at once by drawing a region on the screen. Assuming that users broadcast their contact information (e.g., in the form of a temporary e-mail address), messages could be sent to the selected devices.

Chapter 9

Conclusion

In this thesis, we explored the area of relative positioning using audio signals. Towards this end, we built, presented and evaluated Sonoloc, a mobile app and system that can determine the relative 2D positions of hundreds of mobile devices within audio range with a small, constant number of audio chirps. The system requires devices to have a speaker, microphone, and some sort of internet connectivity, but it does not require any specialized infrastructure. In principle, Sonoloc could also operate without any infrastructure at all, relying on device-to-device network communication instead. Our experimental evaluation, which included extensive simulations and real experiments with over 100 devices in a large lecture hall, shows that Sonoloc can position devices with median/maximal position errors of around 0.06 m and 0.5 m, respectively, under a wide range of conditions.

We designed and parameterized Sonoloc for high worst-case accuracy even under challenging conditions. We built a proof-of-concept prototype that performs well in realistic settings, and we believe that it is a solid step towards accurate, infrastructure-less positioning. Additionally, our evaluation goes far beyond prior work. Still, there are many ways the system could be improved further, including dealing with corner cases (e.g., devices on the same line), dealing with 3D layouts, dealing with sound attenuation/distortion issues (caused by the phones' bodies themselves or by the users' hands), dealing with devices that have high sampling rate mismatch, decreasing the resource consumption and so on. In summary, more evaluation is needed to better understand and then overcome the limits of acoustic positioning systems. In order to improve accuracy and usability, we also need better audio hardware (e.g., more stable sampling clock and ultrasound support). Taking some of these steps is essential for any large-scale deployment of Sonoloc-like relative positioning systems.

Chapter 10

Acknowledgments

I would like to express my sincere gratitude to my PhD supervisor, Peter Druschel for his continuous guidance, valuable feedback, brilliant insights, inspiration and patience. I would also like to thank Bobby Bhattacharjee for all the feedback, support and advice he provided during my studies.

My sincere thanks go also to Nobutaka Ono, who gave me an opportunity to do an internship under his supervision at the National Institute of Informatics in Tokyo. He provided invaluable expertise on various topics related to signal processing, and kindly provided an implementation of the GCC-PHAT and STFT algorithms. Our fruitful research collaboration would not have been possible without him. This internship helped me tremendously to expand my research horizons, and it provided me with an opportunity to interact with Japanese culture meaningfully, which played a very important role in my personal growth.

In addition, I would like to thank:

- Krishna Gummadi and Deepak Garg for being part of my thesis committee and providing insightful and constructive comments during the later stages of my PhD studies
- Trung-Kien Le, who kindly provided us with an implementation of the localization algorithms used by Sonoloc, and worked with us on adapting them to Sonoloc
- Rose Hoberman for her feedback on paper drafts and drafts of this thesis, and for her courses, which taught me valuable skills on several topics ranging from presentation skills to scientific writing
- Carina Schmitt and Christian Klein of our IT staff for their invaluable help in resolving various software, hardware, and infrastructure issues throughout the course of my PhD
- Claudia Richter, Brigitta Hansen and Annika Meiser of our office staff for helping with all sorts of administrative issues
- My fellow students and postdocs at MPI-SWS for creating a motivating environment that enabled me to improve myself both professionally and personally
- My father, Imre Erdélyi, for the continuous support he has given me throughout my time

in graduate school. He encouraged me and gave me valuable advice whenever the need arose. I would not have been able to finish (or even start) my PhD studies without his support.

This work was supported in part by the European Research Council (ERC Synergy imPACT 610150), the German Science Foundation (DFG CRC 1223), the Japan Society for the Promotion of Science (Grant-in-Aid for Scientific Research (A), KAKENHI Grant Number 16H01735), and the National Science Foundation (NSF Awards CNS 1526635 and CNS 1314857).

Chapter 11

Glossary

Audio recording: A list of audio samples recorded by a device's microphone. In the context of Sonoloc, its time interval encompasses a single audio transmission. Due to reflections, multiple signal instances are typically present.

Audio sample: In digital audio systems, analog sound waves are approximated by sampling them at a fixed time interval. The sampling process produces a list of signed numerical values, which are called audio samples. The samples are normalized to some bound, such as +/-1 in the case of floating-point samples.

Autocorrelation: The cross-correlation of a signal with itself. In Sonoloc, autocorrelation is used as a way to evaluate reference signal candidates.

Cross-correlation: A sliding dot product that slides a known pattern (the reference signal) along the samples of the longer, noisy audio recording. For each position, it computes a correlation coefficient that signifies how well the given portion of the noisy audio recording matches the known transmitted signal. The better the match, the higher value the cross-correlation will produce.

Direct path signal: Due to reflections, audio signals travel on multiple paths from transmitter to receiver. The direct path signal refers to the signal that traveled on the shortest, direct path from transmitter to receiver. In an audio recording, the signal instance corresponding to the direct path signal is the earliest one.

Distance difference: Given 3 devices A, B, C, it is the difference $d_{AC} - d_{BC}$ and is denoted by $\tau_{AB|C}$. In the context of Sonoloc, A and B are transmitters, and C is a passive device.

IPDE: Interpoint distance error. For each pair of devices, the absolute difference between the true distance and the distance based on the estimated positions between the devices' centers, in meters. We consider a device's center the average of the coordinates of its speaker and microphones.

Passive device: A device that passively listens to (audio) transmissions from transmitters. It must have at least 1 microphone. A passive device that also has a speaker can be chosen as a transmitter in a later phase of the Sonoloc protocol (as part of the iterative transmitter selection algorithm).

Reference signal: A fixed sequence of audio samples emitted by a transmitter. The sequence is chosen at design time and is optimized for reliable reception. In Sonoloc, the exact sequence is known to both the transmitter and the receiver.

Relative position map: A map that contains a coordinate describing the position of a set of devices relative to other nearby devices. These coordinates are relative in the sense that they are not anchored to any global coordinate system.

RSS: Received Signal Strength, such as that of an incoming Wi-Fi packet.

SNR: Signal-to-noise ratio; in an audio recording, the SNR expresses how strong the signal is compared to the noise.

Sampling rate: The number of audio samples in one second's worth of audio data. It is expressed in Hertz (Hz).

Scalable positioning: In the context of Sonoloc, scalable positioning is defined as an ability to position a large number of devices within acoustic range in a practical way. In Sonoloc's use cases, the number of devices is bounded by the capacity of standard lecture halls, i.e., a few hundred devices.

Signal detection: A signal processing technique to decide whether the reference signal is present in an audio recording. The output is binary: the signal is either detected or not.

Specialized infrastructure: Any infrastructure that goes beyond basic cellular coverage and basic Wi-Fi access. This includes, but is not limited to, requiring explicit pre-deployment effort (e.g., having to measure the positions or transmission power settings of Wi-Fi access points) or requiring the installation of anchor nodes (e.g., ultrasonic room speakers/microphones, modulated artificial lighting, software-defined radios, customized wireless access points).

TDOA: Time difference of arrival; it represents the difference between the arrival times of two (audio) signals. If the time offset between the two transmission timestamps is known, the TDOA can be used to infer the difference in propagation times.

Time delay estimation: A signal processing technique to estimate the offset in an audio recording at which the reference signal starts. It presumes that the *presence* of a signal was already confirmed by signal detection.

Transmitter device: A device that transmits a signal. It must be equipped with at least 1 speaker and 1 microphone.

Bibliography

[1] Paarijaat Aditya et al. “I-Pic: A platform for privacy-compliant image capture”. In: *Proceedings of MobiSys*. 2016. DOI: 10.1145/2906388.2906412.

[2] Teodoro Aguilera et al. “Broadband acoustic local positioning system for mobile devices with multiple access interference cancellation”. In: *Measurement* (2018). DOI: 10.1016/j.measurement.2017.11.046.

[3] T. Akiyama, M. Sugimoto, and H. Hashizume. “Light-synchronized acoustic TOA measurement system for mobile smart nodes”. In: *Proceedings of IPIN*. 2014. DOI: 10.1109/IPIN.2014.7275557.

[4] T. Akiyama, M. Sugimoto, and H. Hashizume. “SyncSync: Time-of-arrival based localization method using light-synchronized acoustic waves for smartphones”. In: *Proceedings of IPIN*. 2015. DOI: 10.1109/IPIN.2015.7346958.

[5] J. B. Allen and L. R. Rabiner. “A unified approach to short-time Fourier analysis and synthesis”. In: *Proceedings of the IEEE* (1977). DOI: 10.1109/PROC.1977.10770.

[6] Wi-Fi Alliance. *Wi-Fi*. URL: https://www.wi-fi.org/ (visited on 09/24/2018).

[7] Plantronics (via Amazon). *Audio 300 PC Mikrofon*. URL: https://www.amazon.de/Plantronics-Audio-300-PC-Mikrofon/dp/B000GCFN72/ (visited on 09/24/2018).

[8] D. Ayllón et al. “Indoor Blind Localization of Smartphones by Means of Sensor Data Fusion”. In: *IEEE Transactions on Instrumentation and Measurement* (2016). DOI: 10.1109/TIM.2015.2494629.

[9] M. Azaria and D. Hertz. “Time delay estimation by generalized cross correlation methods”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1984). DOI: 10.1109/TASSP.1984.1164314.

[10] Nilanjan Banerjee et al. “Virtual Compass: Relative Positioning to Sense Mobile Social Interactions”. In: *Proceedings of Pervasive Computing*. 2010. DOI: 10.1007/978-3-642-12654-3_1.

[11] Leifur Bjornsson. *BLE Beacons for Indoor positioning – Beacon limitations*. URL: https://locatify.com/blog/ble-beacons-no-bull-beacon-review/ (visited on 06/27/2018).

[12] M.S. Brandstein, J.E. Adcock, and H.F. Silverman. “A closed-form location estimator for use with room environment microphone arrays”. In: *IEEE Transactions on Speech Audio Processing* (1997). DOI: 10.1109/89.554268.

[13] G. C. Carter. “Coherence and time delay estimation”. In: *Proceedings of the IEEE* (1987). DOI: 10.1109/PROC.1987.13723.

[14] Ian H Chan. *Swept Sine Chirps for Measuring Impulse Response*. 2010. URL: http://www.thinksrs.com/downloads/PDFs/ApplicationNotes/SR1_SweptSine.pdf (visited on 12/05/2016).

[15] Y.T. Chan and K.C. Ho. “A simple and efficient estimator for hyperbolic location”. In: *IEEE Transactions on Signal Processing* (1994). DOI: 10.1109/78.301830.

[16] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. “Indoor Localization Without the Pain”. In: *Proceedings of MobiCom*. 2010. DOI: 10.1145/1859995.1860016.

[17] C. E. Cook. “Pulse Compression-Key to More Efficient Radar Transmission”. In: *Proceedings of the IRE* (1960). DOI: 10.1109/JRPROC.1960.287599.

[18] Charles Cook. *Radar Signals: An Introduction to Theory and Application*. Electrical science series. Elsevier Science, 2012. ISBN: 9780323146302.

[19] Frank Dabek et al. “Vivaldi: A Decentralized Network Coordinate System”. In: *Proceedings of SIGCOMM*. 2004. DOI: 10.1145/1015467.1015471.

[20] D. Dardari et al. “Ranging With Ultrawide Bandwidth Signals in Multipath Environments”. In: *Proceedings of the IEEE* (2009). DOI: 10.1109/JPROC.2008.2008846.

[21] Dynacord. *DSA Series Power Amplifiers*. URL: https://www.dynacord.com/product-family.php?id=259&language=en (visited on 09/24/2018).

[22] Ramsey M. Faragher and Robert K. Harle. “Towards an Efficient, Intelligent, Opportunistic Smartphone Indoor Positioning System”. In: *Journal of the Institute of Navigation* (2015). DOI: 10.1002/navi.76.

[23] Raspberry Pi Foundation. *Raspberry Pi 3 Model B*. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ (visited on 09/24/2018).

[24] S.G. Glisic and P.A. Leppänen. *Code Division Multiple Access Communications*. Springer, 2012. ISBN: 9781461522515. DOI: 10.1007/978-1-4615-2251-5.

[25] S.W. Golomb. *Shift register sequences*. Aegean Park Press, 1981. ISBN: 0894120484.

[26] S.W. Golomb and G. Gong. *Signal Design for Good Correlation: For Wireless Communication, Cryptography, and Radar*. Cambridge University Press, 2005. ISBN: 9780521821049.

[27] GSMArena. *Motorola Nexus 6 specifications*. URL: https://www.gsmarena.com/motorola_nexus_6-6604.php (visited on 09/24/2018).

[28] Hiromichi Hashizume et al. “Fast and Accurate Positioning Technique Using Ultrasonic Phase Accordance Method”. In: *Proceedings of IEEE TENCON*. 2005. DOI: 10.1109/TENCON.2005.301009.

[29] Jorge Herrera and Hyung-Suk Kim. “Ping-pong: Using smartphones to measure distances and relative positions”. In: *The Journal of the Acoustical Society of America* (2013). DOI: 10.1121/1.4831185.

[30] London Trust Media Inc. *Private Internet Access | Anonymous VPN Service Provider*. URL: https://www.privateinternetaccess.com/ (visited on 09/24/2018).

[31] Sony Mobile Communications Inc. *Xperia XZ1 Compact specifications*. URL: https://www.sonymobile.com/global-en/products/phones/xperia-xz1-compact/specifications/ (visited on 09/24/2018).

[32] The MathWorks Inc. *Constant False Alarm Rate (CFAR) Detection*. URL: https://www.mathworks.com/help/phased/examples/constant-false-alarm-rate-cfar-detection.html (visited on 09/28/2017).

[33] The MathWorks Inc. *Fourier Transforms*. URL: https://de.mathworks.com/help/matlab/math/fourier-transforms.html (visited on 09/25/2018).

[34] The MathWorks Inc. *Multidimensional Scaling*. URL: https://www.mathworks.com/help/stats/multidimensional-scaling.html (visited on 01/21/2018).

[35] The MathWorks Inc. *Singular Values*. URL: https://de.mathworks.com/help/matlab/math/singular-values.html (visited on 09/25/2018).

[36] TunnelBear Inc. *TunnelBear: Secure VPN Service*. URL: https://www.tunnelbear.com/ (visited on 09/28/2018).

[37] C. Knapp and G. Carter. “The generalized correlation method for estimation of time delay”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1976). DOI: 10.1109/TASSP.1976.1162830.

[38] Manikanta Kotaru et al. “SpotFi: Decimeter Level Localization Using WiFi”. In: *Proceedings of SIGCOMM*. 2015. DOI: 10.1145/2785956.2787487.

[39] Swarun Kumar et al. “Accurate Indoor Localization With Zero Start-up Cost”. In: *Proceedings of Mobicom*. 2014. DOI: 10.1145/2639108.2639142.

[40] Patrick Lazik and Anthony Rowe. “Indoor Pseudo-ranging of Mobile Devices Using Ultrasonic Chirps”. In: *Proceedings of SenSys*. 2012. DOI: 10.1145/2426656.2426667.

[41] Patrick Lazik et al. “ALPS: A Bluetooth and Ultrasound Platform for Mapping and Localization”. In: *Proceedings of SenSys*. 2015. DOI: 10.1145/2809695.2809727.

[42] T. K. Le and N. Ono. “Closed-form and Near closed-form solutions for TDOA-based joint source and sensor localization”. In: *IEEE Transactions on Signal Processing* (2017). DOI: 10.1109/TSP.2016.2633784.

[43] T.-K. Le and N. Ono. “Closed-form and Near closed-form solutions for TOA-based joint source and sensor localization”. In: *IEEE Transactions on Signal Processing* (2016). DOI: 10.1109/TSP.2016.2569465.

[44] T.-K. Le and N. Ono. “Numerical Formulae for TOA-based Microphone and Source Localization”. In: *Proceedings of IWAENC*. 2014. DOI: 10.1109/IWAENC.2014.6954002.

[45] T.-K. Le and N. Ono. “Robust TDOA-based Joint Source and Microphone Localization in Reverberant Environment using Medians of Acceptable Recovered TOAs”. In: *Proceedings of IWAENC*. 2016. DOI: 10.1109/IWAENC.2016.7602942.

[46] T.-K. Le et al. “Experimental validation of TOA-based methods for microphones array positions calibration”. In: *Proceedings of ICASSP*. 2016. DOI: 10.1109/ICASSP.2016.7472271.

[47] Trung-Kien Le and Nobutaka Ono. “Closed-form solution for TDOA-based joint source and sensor localization in two-dimensional space”. In: *Proceedings of EUSIPCO*. 2016. DOI: 10.1109/EUSIPCO.2016.7760473.

[48] Matthew Lentz et al. “SDDR: Light-Weight, Secure Mobile Encounters”. In: *Proceedings of USENIX Security*. 2014. ISBN: 978-1-931971-15-7. URL: https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/lentz.

[49] Nadav Levanon and Eli Mozeson. *Radar signals*. John Wiley & Sons, 2004. ISBN: 978-0-471-47378-7.

[50] Hongbo Liu et al. “Push the Limit of WiFi Based Localization for Smartphones”. In: *Proceedings of MobiCom*. 2012. DOI: 10.1145/2348543.2348581.

[51] Kaikai Liu, Xinxin Liu, and Xiaolin Li. “Guoguo: Enabling Fine-grained Indoor Localization via Smartphone”. In: *Proceeding of MobiSys*. 2013. DOI: 10.1145/2462456.2464450.

[52] Kaikai Liu et al. “Towards accurate acoustic localization on a smartphone”. In: *Proceedings of INFOCOM*. 2013. DOI: 10.1109/INFCOM.2013.6566822.

[53] inMusicBrands LLC. *Alesis RA150 Amplifier*. URL: https://www.alesis.com/products/legacy/ra150 (visited on 09/24/2018).

[54] Locatify. *In Practice: Precise Indoor Location Detection with UWB / Ultra-Wideband*. URL: https://locatify.com/blog/in-practice-precise-indoor-location-detection-with-uwb-ultra-wideband/ (visited on 06/27/2018).

[55] Sérgio I. Lopes et al. “Accurate smartphone indoor positioning using a WSN infrastructure and non-invasive audio for TDoA estimation”. In: *Pervasive and Mobile Computing* (2015). DOI: 10.1016/j.pmcj.2014.09.003.

[56] Dimitrios Lymberopoulos et al. “A Realistic Evaluation and Comparison of Indoor Location Technologies: Experiences and Lessons Learned”. In: *Proceedings of IPSN*. 2015. DOI: 10.1145/2737095.2737726.

[57] PeakTech Prüf- und Messtechnik GmbH. *PeakTech 5055 Digital Sound Level Meter*. URL: https://www.peaktech.de/productdetail/kategorie/schallpegelmessgeraete/produkt/p-5055.html (visited on 09/24/2018).

[58] J.N. Moutinho, R.E. Araújo, and D. Freitas. “Indoor localization with audible sound — Towards practical implementation”. In: *Pervasive and Mobile Computing* (2016). DOI: 10.1016/j.pmcj.2015.10.016.

[59] Rajalakshmi Nandakumar, Krishna Kant Chintalapudi, and Venkata N. Padmanabhan. “Centaur: Locating Devices in an Office Environment”. In: *Proceedings of MobiCom*. 2012. DOI: 10.1145/2348543.2348579.

[60] National Coordination Office for Space-Based Positioning, Navigation, and Timing. *GPS: The Global Positioning System*. URL: https://www.gps.gov/ (visited on 09/25/2018).

[61] J. Neyman and E. S. Pearson. “On the Problem of the Most Efficient Tests of Statistical Hypotheses”. In: *Philosophical Transactions of the Royal Society* (1933). DOI: 10.1098/rsta.1933.0009.

[62] NordVPN. *NordVPN Official Website*. URL: https://nordvpn.com/home-app/ (visited on 09/24/2018).

[63] N. Ono, K. Shibata, and H. Kameoka. “Self-localization and channel synchronization of smartphone arrays using sound emissions”. In: *Proceedings of APSIPA*. 2016. DOI: 10.1109/APSIPA.2016.7820778.

[64] N. Ono et al. “Blind alignment of asynchronously recorded signals for distributed microphone array”. In: *Proceedings of WASPAA*. 2009. DOI: 10.1109/ASPAA.2009.5346505.

[65] Chunyi Peng et al. “BeepBeep: A High Accuracy Acoustic Ranging System Using COTS Mobile Devices”. In: *Proceedings of SenSys*. 2007. DOI: 10.1145/1322263.1322265.

[66] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. “The Cricket Location-support System”. In: *Proceedings of MobiCom*. 2000. DOI: 10.1145/345910.345917.

[67] JBL Professional. *CBT 100LA-1 Constant Beamwidth Technology Line Array Column Loudspeaker*. URL: http://www.jblpro.com/www/products/installed-sound/cbt-series/cbt100la-1#.W6jrl_axURQ (visited on 09/24/2018).

[68] JBL Professional. *Control 1 Pro*. URL: http://www.jblpro.com/www/products/recording-broadcast/control-1-pro#.W6jnmfaxURQ (visited on 09/24/2018).

[69] JBL Professional. *Control 24CT Background/Foreground Ceiling Loudspeakers*. URL: http://www.jblpro.com/www/products/installed-sound/control-contractor-series/control-24ct#.W6jstvaxURR (visited on 09/24/2018).

[70] Tor Project. *Tor Project | Privacy Online*. URL: https://www.torproject.org/ (visited on 09/24/2018).

[71] Jian Qiu et al. "On the Feasibility of Real-time Phone-to-phone 3D Localization". In: *Proceedings of SenSys*. 2011. DOI: 10.1145/2070942.2070962.

[72] Jun-Wei Qiu et al. "A D2D relative positioning system on smart devices". In: *WCNC*. 2014. DOI: 10.1109/WCNC.2014.6952645.

[73] Sabrent. *AU-MMSA USB External Stereo 3D Sound Adapter*. URL: https://www.sabrent.com/product/AU-MMSA/usb-external-stereo-3d-sound-adapter-black/ (visited on 09/24/2018).

[74] Occupational Safety and US Department of Labor Health Administration. *Occupational noise exposure*. URL: https://www.osha.gov/pls/oshaweb/owadisp.show_document?p_table=STANDARDS&p_id=9735 (visited on 06/20/2018).

[75] T. Sathyan, D. Humphrey, and M. Hedley. "WASP: A System and Algorithms for Accurate Radio Localization Using Low-Cost Hardware". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* (2011). DOI: 10.1109/TSMCC.2010.2051027.

[76] H.C. Schau and A.Z. Robinson. "Passive source localization employing intersecting spherical surfaces from time-of-arrival differences". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1987). DOI: 10.1109/TASSP.1987.1165266.

[77] Souvik Sen et al. "Avoiding Multipath to Revive Inbuilding WiFi Localization". In: *Proceedings of MobiSys*. 2013. DOI: 10.1145/2462456.2464463.

[78] Bluetooth SIG. *Bluetooth Core Specifications, Volume 6 (Low Energy Controller Volume)*. URL: https://www.bluetooth.com/specifications/bluetooth-core-specification (visited on 09/24/2018).

[79] J. Smith and J. Abel. "Closed-form least squares source location estimation from range-difference measurements". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1987). DOI: 10.1109/TASSP.1987.1165089.

[80] P. Stoica and J. Li. "Lecture Notes - Source Localization from Range-Difference Measurements". In: *IEEE Signal Processing Magazine* (2006). DOI: 10.1109/SP-M.2006.248717.

[81] Paul Syverson, R Dingledine, and N Mathewson. "Tor: The second-generation onion router". In: *Proceedings of Usenix Security*. 2004.

[82] Warren S Torgerson. "Multidimensional scaling: I. Theory and method". In: *Psychometrika* (1952). DOI: 10.1007/BF02288916.

[83] Jie Xiong and Kyle Jamieson. "ArrayTrack: A Fine-Grained Indoor Location System". In: *Proceedings of NSDI*. 2013. ISBN: 978-1-931971-00-3. URL: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/xiong.

[84] Jie Xiong, Kyle Jamieson, and Karthikeyan Sundaresan. "Synchronicity: Pushing the Envelope of Fine-Grained Localization with Distributed MIMO". In: *Proceedings of HotWireless*. 2014. DOI: 10.1145/2643614.2643619.

[85] Jie Xiong, Karthikeyan Sundaresan, and Kyle Jamieson. "ToneTrack: Leveraging Frequency-Agile Radios for Time-Based Indoor Wireless Localization". In: *Proceedings of Mobicom*. 2015. DOI: 10.1145/2789168.2790125.

[86] L. Yang and K.C. Ho. "An approximately efficient TDOA localization algorithm in closed-form for locating multiple disjoint sources with erroneous sensor positions". In: *IEEE Transactions on Signal Processing* (2009). DOI: 10.1109/TSP.2009.2027765.

[87] Gale Young and Alston S Householder. "Discussion of a set of points in terms of their mutual distances". In: *Psychometrika* (1938). DOI: 10.1007/BF02287916.

[88] Moustafa Youssef and Ashok Agrawala. "The Horus WLAN Location Determination System". In: *Proceedings of MobiSys*. 2005. DOI: 10.1145/1067170.1067193.

[89] Moustafa Youssef et al. "PinPoint: An Asynchronous Time-Based Location Determination System". In: *Proceedings of MobiSys*. 2006. DOI: 10.1145/1134680.1134698.

[90] Cha Zhang, D. Florencio, and Zhengyou Zhang. "Why does PHAT work well in low noise, reverberative environments?" In: *Proceedings of ICASSP*. 2008. DOI: 10.1109/ICASSP.2008.4518172.

[91] Zengbin Zhang et al. "SwordFight: Enabling a New Class of Phone-to-phone Action Games on Commodity Phones". In: *Proceedings of MobiSys*. 2012. DOI: 10.1145/2307636.2307638.

[92] Neal Zierler. "Linear recurring sequences". In: *Journal of the Society for Industrial and Applied Mathematics* (1959). DOI: 10.1137/0107003.