# Estimation of Internal Delays

BSc Thesis

Tim Ammerlaan & Tim Al

Delft University of Technology

TU Delft

# Estimation of Internal Delays

## BSc Thesis

by Tim Ammerlaan and Tim Al

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended on Monday July 2, 2018 at 1:00 PM.


| | | |
|---|---|---|
| Student numbers: | 4472969, 4439848 | |
| Project duration: | April 23, 2018 – June 18, 2018 | |
| Thesis committee: | Dr. Ir. R.C. Hendriks, | TU Delft, supervisor |
| | Dr. Ir. R. Heusdens, | TU Delft, supervisor |
| | Dr. J. Martinez, | TU Delft, supervisor |
| | Dr. O. A. Krasnov | TU Delft |
| | Dr. Ir. B.J. Kooij | TU Delft |


TU Delft

# Preface

This thesis adresses the problem on the estimation of internal delays, it describes an implemented algorithm which determines both onset times and internal delays of microphones in a wireless oriented network. This thesis is written to fulfill the graduation requirements of the Bachelor of Electrical Engineering at the TU Delft. The thesis describes the research and implementation during the period April to June 2018.

The project was at request of Bosch Security and Safety systems Eindhoven, with the supervision of Hans van der Schaar. During the research some problems were faced, the initial division of work between the groups proved to be difficult due to different circumstances, for quite some time it was not known if we could access the software stack of Bosch. This brought along some delay resulting in only an initial implementation of the algorithm in estimating internal delays.

Great thanks go out to the supervisors of the project for their guidance and support during the project. Furthermore appreciation goes to the fellow group members for this bachelor graduation project, for their help and the good ambiance that was created with the group as a whole.

*Tim Ammerlaan and Tim Al*
*Delft, June 2018*

# Abstract

In this thesis report, the design of the estimation of internal delays within speakers and microphones will be covered. The estimation is done via an iterational algorithm which converges to the different delays. The design choice of the estimation of those delays follows from an initial attempt to improve synchronization of the Bosch DICENTIS microphone units. With an unit being a system is placed on the desk of an attendee of a conference, consisting of a microphone and a speaker. After coming to the conclusion that the current level of synchronization already sufficed the focus was shifted towards the estimation of the internal delays.

The choice of algorithm that was used for the estimation is covered in the in the state of the art analysis of the current internal delay estimation techniques. The subsystem receives pre-determined times between units and uses these to estimate the internal delays. This estimation is done with a random initialization of the delays (within reasonable margins for the delays). After which this estimation converges towards the real values by minimizing a Frobenius norm between the rank three approximation and the received times. This is elaborated on in the Estimating Internal Delays section. The algorithm can also make use of a regularization term which decreases the time required for the estimation of the delays. The results of the algorithm are discussed in the Implementation section, which consists of a number of MATLAB simulations using the implemented algorithm. Using the results, a conclusion is drawn for the viability of the solution after which a recommendation of future work is given.

# Contents

# 1 Introduction

## 1.1. Problem Definition

Audio localization has become more prominent during the last few years. This due to the recent development in wireless devices. One of which is the localization of microphones that are used during conferences. For example cameras have been introduced in large conferences during the last few years, this was introduced in order to make the conferences more interactive. Using these camera's the fellow participants of the conference are able to see the various speakers and feel more engaged during the conference. In order to do this, localization of the different conference members is necessary. With a setup with different microphones this will result in the localization of the microphones of the different attendees of the conference.

Bosch security systems provides these conferences with microphone units in order to be able to have a clear conversation with everyone in the room. Examples of conferences of such scale are conferences of the European Parliament where over 700 attendees can be. To ensure the interactive element of a camera pointing towards the speaker in action localization of each of these microphones is essential. Locating these units by hand would be impossible to do in a short time frame, therefor an automated process is desired.

The system that is used is the Bosch DICENTIS system which can be seen in Figure 1.1. The system consist of a wireless access point and a number of wireless microphone units. Each of these units has a receiver (a microphone) and a sender(a speaker). These parts will are used to determine the location of each of these units by generating audio pulses on one device which is received by the rest of the devices.

Figure 1.1: A conference system unit from Bosch which should be able to comply with the implemented algorithm

### Situation Assessment

Up till now the localization of the different microphones is done by hand. The microphones are distributed before a conference after which someone will calibrate the system. During this calibration a camera is pointed to all the positions of the different microphones and the exact location of each microphone is saved. Since this process requires someone to be locating the different positions this is both a time consuming and expensive job.

In order to diminish the cost and the time the calibrating consumes an automated algorithm is required. By automating the process the localization should be quicker and more consistent. This algorithm is split up in three parts, determining distances between nodes, estimating the onset times and the internal delays of the microphones and lastly converting the corrected distances to a coordinate system.
The estimation of the onset times and the internal delays of the different microphones will be essential to

accurately determine the locations of the Bosch units. This due to the accumulation of the errors in the estimation, every unaccounted time delay will cause a corresponding distance error. The estimation of these delays is the main focus of this thesis.

### Scoping and Bounding

The localization algorithm that is developed is a standalone product for a variety of microphone setups. The algorithm works with combined microphone and speaker units or with a single sound source with multiple sound events in different positions. The can be used on the Bosch DICENTIS system, which consists of a wireless access point and a number of wireless microphone units. Since the algorithm is designed for non co-located sources, speakers, and receivers, microphones, the algorithm can be used on different setups. Two of which are depend on the microphone on the wireless units. These can be seen in Figure 1.1 and Figure 1.2. For the unit with the fixed microphone (1.1) co location could still be a viable option, but for the unit with the stem microphone a non co located solution has to be found.

The project is divided in 3 sub groups, each with their own deliverable. Combining these deliverables result in a functioning final product. The three subgroups are, Distance estimation, Internal delay estimation and Localization. The distance estimation group determines distances between different sets of microphones. This is done by producing acoustic signals and comparing the time between transmitting and receiving between two units. This distance can be converted to time by dividing by the speed of the acoustic signal in the medium.

Figure 1.2: A conference system unit from Bosch with a stem microphone

The internal delay estimation sub group receives these times and performs an error correction on them. This because there are some inaccuracies in those times. For wireless systems, the system clocks may not be synchronized completely, this results in a difference in local time and thus a difference in received time. Furthermore there are also a number of unknown delays between the moment that a signal is 'sent' and when it is 'received'. The model that is used is discussed in chapter 3.
The localization subgroup is responsible for the coordinate localization. This group determines the exact locations based on the received times of each of the microphones.
Initially the aim of the internal delay estimation subgroup was to synchronize the clocks of the system. However due to the presence of the synchronized system of Bosch (which is synchronized up to a maximum error of 100 microseconds) the immediate need for an extra optimization vanished. The importance of the undetermined internal delays was still a valid problem, causing the focus to shift towards these delays. This can be illustrated by the error in the localization in case of the current level of synchronization. The current level of synchronization would cause an error of $343 \cdot 1 \times 10^{-4} = 3.43$cm while an internal delay of the order $1 \times 10^{-3}$ would cause an error of 34.3cm. This confirms the importance of the estimation of the internal delays. The overview of this project can be seen in Figure 1.3

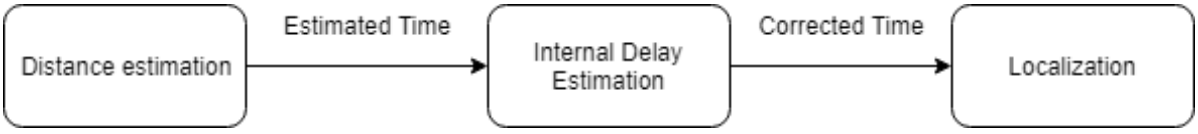


Figure 1.3: The block diagram of the project

## 1.2. Analysis of ranging techniques

A well known localization method that is used everyday is the global positioning system (GPS). However due to the fact that GPS sensors are expensive, have a high power consumption and obtain poor accuracy indoors

[1] they are not suited for localization in indoor areas.

To solve this, research has been done to find alternatives to GPS, these methods use different media. A lot of research has been done to find different methods to localize sensors. Methods using different media such as radio frequent and audio have been developed to perform the localization [2]. The use of radio frequency signals require a strict synchronization [3].

To estimate the distances there are several techniques, the received signal strength can be used to determine the relative distance between two nodes in a network [4], other techniques can depend on the Angle of Arrival (AOA) [5]. Or techniques which are time based, they make use of the Time Of Arrival (TOA) [6][7] or the Time Difference On Arrival (TDOA) [8][9]. The Bosch DICENTIS microphone units can be located using the standard WIFI module or making use of the speakers and microphones. For this project a time-based localization algorithm is used. To ensure that the estimated locations are as accurate as possible the Time Of Flight should only be taken into account. Thus the other undetermined time between sending and receiving the signal should be removed from the measured TOF. This results in the estimation of the delays between the moment of sending and receiving. The unknown delays of the different senders can be circumvented by a TDOA algorithm [10]. However, internal delays on the receiving side will still have an effect on the estimated time of flight.

In [11] a method using radio frequencies is described which compensates for both clock offset and frequency offsets. In [12] a processes of different iterations is used to converge the estimation of the delays to the correct values while in [13] a data fitting technique which is based on structured total least squares is used [14].

## 1.3. Document Structure

The structure of this report is as follows, in Chapter 2 the requirements for the complete localization system that are imposed by Bosch will be covered. Furthermore it covers the assumptions that could be made according to Bosch. Based on these requirements and assumptions, the relevant requirements will for the estimating delays part of the project are described. Chapter 3 contains an elaboration on the design process. This chapter will justify the choice of algorithm to compute the internal delays and describe the different steps of the algorithm. The results of the simulated algorithm are described in the Chapter 4. In Chapter 5 the results of the simulation are discussed. Lastly the conclusions that could be drawn and recommendations for future work are described in Chapter 6.

# 2

# Program of Requirements

## 2.1. Project Based Program Of Requirements

The project based program of requirements are the requirements and assumptions that are imposed over the overlapping BAP group project. This concerns the system that can be presented to project supplier, which is Bosch.

### Assumptions

When discussing the system requirements with the contact person at Bosch, there were a set of assumptions that were allowed to be made. Furthermore, our group also added a set of assumptions based on things that were not discussed at the meeting with the Bosch contact person. The made assumptions for the overarching project are as follows:

1. As indicated by the representative at Bosch when discussing the system requirements, there is line of sight between the Bosch DICENTIS units and between any given unit and the access point.
2. Minimal distance between units is 75 cm.
3. Microphones on the Bosch DICENTIS units can accurately capture audio from a distance of at least 30 m.
4. There is at least one unit per 15 $m^2$.
5. It is possible to upgrade the existing software on the Bosch DICENTIS system to accommodate the to be designed localization system functionalities.
6. It is assumed that the runtime of a MATLAB program run on system with recommended requirements is comparable to the runtime of the same program written for the Bosch DICENTIS system.

### Mandatory requirements

These are criteria of which the system should always, at the very least, comply with. These can be subdivided into functional and non-functional requirements. Functional requirements being requirements of what the system must do, and non-functional requirements being attributes that the system must have.

1. **Non-Functional requirements**
   (a) The localization system must be within 10 cm accurate.
   (b) The localization system must be scalable up to 120 units.
   (c) A 3D localization method is necessary.
   (d) The localization should work in conference rooms with dimensions up to 30 × 30 m.
2. **Functional requirements**
   (a) The localization speed should have a maximum duration of 15 minutes for systems of more than 100 units.

(b) The localization speed should have a maximum duration of 5 minutes for systems of less than 20 units.

(c) The localization algorithm has to comply with the current Bosch DICENTIS conference system hardware characteristics.

(d) The localization procedure can only be initiated remote interface, so no one accidentally starts it during a conference.

(e) The system as a whole should not exceed the maximum allowed sound pressure level.

(f) The addition to the system of Bosch should not bypass any safety precautions taken by Bosch.

### Trade-off requirements

These are criteria of which it is preferable to comply with as much as possible:

1. Minimize the number of manually configured anchor points.
2. Minimize the number of units required for the system to work.
3. The localization should work with the least possible additional hardware.
4. The localization speed should be as fast as possible.
5. The localization should be as accurate as possible.
6. A purely 2D localization method is usable besides a 3D localization method.

## 2.2. Subgroup Based Program Of Requirements

With respect to the work being done on the part of internal delay and onset time estimation, a few requirements are lit out.

### Accuracy

The localization system must be within 10cm accurate. Given that it is expected that the largest error is made while doing the real measurements and the estimation of the locations, the error introduced by not taking internal delays into account must be below 2cm. This results with a speed of sound of 343m/s at an error in timing of at most 58$\mu$s.

### Scalability and timing

The localization has limits with respect to the maximum duration in computation time. For small setups of less than 20 units the total time must not exceed the 5 minutes, and for larges setups of more than 100 units the localization must not take more than 15 minutes. For the part on internal delay and onset time estimation, these limits for the algorithm are set to 1 and 3 minutes respectively.

### Maximum dimensions

The requirements with respect to the maximum dimensions is for the part of internal delay and onset time estimation not a requirement which is something that will result in large errors.

### Assumptions

During the design process, the following assumptions are made that are relevant to the delay estimation part.

1. Minimal distance between units is 75cm.
2. Microphones on the Bosch DICENTIS units can accurately capture audio from a distance of at least 30m.
3. It is assumed that the runtime of a MATLAB program run on system with recommended requirements is comparable to the runtime of the same program written for the Bosch DICENTIS system.
4. The internal delays of the units are in the range of 0 up to 20ms and the onset times vary between the 0 and 30ms.
5. The clocks of the different units are all synchronized within 100$\mu$s.

# 3

# Estimating internal delays

This chapter will justify the choice of algorithm used to determine the onset time and internal delay of the different microphones. Furthermore the theory behind the implementation of the algorithm is explained. It will describe the solution which is found, both with and without the regularization term lambda.

## 3.1. Model used for estimating delays

When assumed that the local clocks of the sending and receiving end are synchronized, the time it takes before the pulse is detected at the receiving end is not only the Time of Flight, which depends on the physical distance between the points. There is namely a unknown delay before a pulse is actually send and a delay before the pulse is actually detected. These are the onset time $\tau$ and the internal delay $\delta$ respectively. When not taking these two delays into account when estimating a distance between two points result in a wrong estimation of the distance between them. This becomes a serious problem when the to be estimated Time of Flight is short and the delays can not be neglected anymore, which is the case when a short distance is to be estimated.
The goal is to estimate the onset times and the internal delays, to get a better estimation of the Time of Flight which will then result in a better localization. Starting from the measured Times of Arrival, the onset times and internal delays will be estimated where after a set of corrected relative Times of Arrival are being passed through. The model used to estimate the onset times and internal delays can be seen in Figure 3.1.

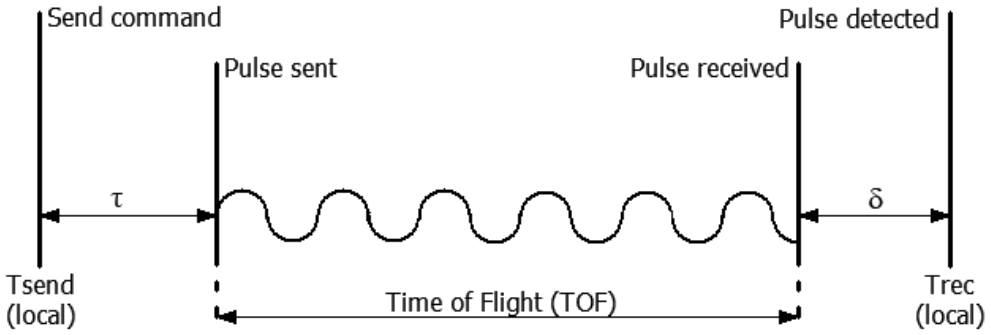


Figure 3.1: The model which is used to determine the onset times and internal delays

## 3.2. Definition of Notation

With respect to notation, in the upcoming sections most is explained when actually used. Furthermore, there are a few general things which are summarized below;
$\|\cdot\|$ denotes 2-norm,
$\|\cdot\|_F$ denotes the Frobenius norm,
$[\mathbf{A}]^T$ denotes the transpose of $\mathbf{A}$,
$\mathbf{B}^{(n)}$ denotes the matrix or vector $\mathbf{B}$ at iteration $n$ and
$\hat{\mathbf{d}}$ denotes an estimator of the real value $\mathbf{d}$.
$\bar{\mathbf{E}}$ denotes an estimation of $\mathbf{E}$.
With a *unit* a node in the network is meant, which consist of a speaker (sender) and microphone (receiver). But the sender and receiver are not necessarily at the same physical location.

## 3.3. Problem Formulation

The problem of the localization of the microphones can be written as in (3.1). The measured times $t_{ij}$ will consist of the time of flight between sender $i$ ($r_i$) and receiver $j$ ($s_j$), this can be converted to the desired distance by dividing it by the speed of sound in the medium between the sender and receiver ($c$). The $\tau_j$ represents the onset time of the sender, which is the time it takes between giving a command to send and actually sending the acoustic wave. The internal delay of the receiver is represented by $\delta_i$, this is the time it takes before the pulse is recognized after receiving it. Measurement noise is represented with $v_{ij}$.

$$t_{ij} = \frac{||\mathbf{r}_i - \mathbf{s}_j||}{c} + \tau_j + \delta_i + v_{ij} \tag{3.1}$$

The next step is to rewrite the equation above. Thereby the speed of sound $c$ is set to 1 and the assumption is made that there is no measurement noise and $v_{ij} = 0$. After expanding the norm of the distance in (3.1) (3.2) is derived.

$$r_i^T r_i + s_j^T s_j - 2r_i^T s_j = t_{ij}^2 + \tau_j^2 + \delta_i^2 - 2(t_{ij}\tau_j + t_{ij}\delta_i - \delta_i\tau_j) \tag{3.2}$$

In the used algorithm receiver $i = 1$ is chosen as a reference node. By subtracting the equation for $i = 1$ from (3.2) above, this results in (3.3).

$$\begin{aligned} r_i^T r_i - r_1^T r_1 - 2(r_i - r_1)^T s_j = {} & t_{ij}^2 - t_{1j}^2 + \delta_i^2 - \delta_1^2 \\ & - 2t_{ij}(\delta_i + \tau_j) + 2t_{1j}(\delta_1 + \tau_j) \\ & + 2\tau_j(\delta_i - \delta_1) \end{aligned} \tag{3.3}$$

As another reference point on the sending side, sender $j = 1$ is chosen as a reference. The time reference for the acoustic events can be determined with respect to one reference point, thereby setting $\tau_1 = 0$. When subtracting the equation for $j = 1$ from (3.3), this results in (3.4).

$$\begin{aligned} -2(r_i - r_1)^T(s_i - s_1) \quad = \quad & t_{ij}^2 - t_{1j}^2 - t_{i1}^2 + t_{11}^2 \\ & - 2\delta_i(t_{ij} - t_{i1}) + 2\delta_1(t_{1j} - t_{11}) \\ & - 2\tau_j(t_{ij} - t_{1j}) + 2\tau_j(\delta_i - \delta_1) \end{aligned} \tag{3.4}$$

## 3.4. Delay estimation using matrices

The following step in the procedure to solve to problem is writing the equations in matrix form. Given (3.4), this can be written as in (3.6). On the left side of the equal sign the location matrices are stated, where $\bar{\mathbf{R}}$ is an $(I-1)\times 3$ location matrix of the receivers and $\bar{\mathbf{S}}$ is an $(J-1)\times 3$ location matrix of the senders. The total number of receivers is equal to $I$ and the total number of transmitters is represented by $J$. On the right side of the equal sign of (3.6) $\mathbf{T}$ represents the squared terms of $t$ from (3.4), where $\mathbf{T}_{ij}$ is $t_{ij}^2 - t_{1j}^2 - t_{i1}^2 - t_{11}^2$ for $i = 2,...,I$ and $j = 2,...,J$.
The remaining parts of (3.4) are represented in $\Gamma_{ij}$ and $\mathbf{A}(\mathbf{p})$. Where $\Gamma_{ij}$ represents the term $2(\delta_i\text{-}\delta_1)\tau_j$. $\mathbf{A}(\mathbf{p})$ is a matrix composed of $C^{-1}_{I-1\times J-1}(\mathbf{Wp})$, where the C-operation transforms a matrix into a column vector and

the inverse operator ($C^{-1}_{M\times N}$) transforms a vector to a $M\times N$ matrix. $\mathbf{W}$ is a $(I-1)(J-1)\times(I+J-1)$ matrix which consists of $(t_{ij}-t_{i1})$, $(t_{1j}-t_{11})$ and $(t_{ij}-t_{1j})$. $\mathbf{p}$ is a vector of length $I+J-1$ which consist of the unknown internal delays ($\delta_i$) and the unknown onset times ($\tau_j$). $\mathbf{p}$ is build up according to the form as in (3.5).

$$\mathbf{p} = \begin{bmatrix}\delta_1, & \delta_2, & \dots & \delta_I, & \tau_2, & \tau_3, & \dots & \tau_J\end{bmatrix}^T \tag{3.5}$$

Hence, after taking the product of $\mathbf{Wp}$ the result is a vector, note the need of the inverse C-operation before adding $\mathbf{T}$ and $\Gamma$ which are of size $(I-1)\times(J-1)$.

$$-2\bar{\mathbf{R}}\bar{\mathbf{S}}^T = \mathbf{T} + \mathbf{A}(\mathbf{p}) + \Gamma \tag{3.6}$$

## 3.5. Recursively estimating delays

As can be seen in (3.6), $\Gamma$ and $\mathbf{A}(\mathbf{p})$ are matrices that act as a correction for the internal delays and onset times on $\mathbf{T}$. To determine the values of the unknowns in $\mathbf{p}$ from (3.6) an iterative approach is used. First an estimator $\hat{\mathbf{p}}$ is introduced, which is initialized randomly. This results in an estimation of corrected times $\hat{\mathbf{T}}$, as can be seen in (3.7). Due to the estimator of the unknowns in $\hat{\mathbf{p}}$, the estimated versions of $\mathbf{A}(\mathbf{p})$ and $\Gamma$ become $\mathbf{A}(\hat{\mathbf{p}})$ and $\hat{\Gamma}$ respectively.

$$\hat{\mathbf{T}} = \mathbf{T} + \mathbf{A}(\hat{\mathbf{p}}) + \hat{\Gamma} \tag{3.7}$$

When looking carefully to the left side of the equal sign in (3.6) the product $-2\bar{\mathbf{R}}\bar{\mathbf{S}}^T$ is of rank 3, the $x$, $y$ and $z$ in Cartesian coordinates. This is true for the case that there are three or more microphones used. However, on the right side of (3.6) for which the estimator $\hat{\mathbf{T}}$ is used, it also has to be of rank three. This results in a rank reduction problem for $\hat{\mathbf{T}}$ and thus $\hat{\mathbf{p}}$, which is solved by making use of the Eckart-Young low-rank approximation theorem [15]. This theorem makes use of the Singular Value Decomposition (SVD) to make the best rank-$r$ approximation which is given by (3.8).

$$\tilde{\mathbf{X}} = \mathbf{U}\tilde{\Sigma}_r\mathbf{V}^T \tag{3.8}$$

In (3.8) $\tilde{\mathbf{X}}$ is the best rank-$r$ approximation of $\mathbf{X}$ and $\tilde{\Sigma_r}$ only consists of the largest $r$ singular values on the diagonal for a rank-$r$ approximation. The other values in the matrix $\tilde{\Sigma_r}$ are set to zero. Matrix $\mathbf{U}$ is a $(I-1)\times(I-1)$ matrix that will be truncated to $(I-1)\times r$ and matrix $\mathbf{V}$ is after truncation a matrix of size $(J-1)\times r$. [16] To find the best fitting values of the estimator of $\hat{\mathbf{p}}$, the values are after their random initialization updated every iteration by minimizing the cost function which can be seen in (3.9).

$$\hat{\mathbf{p}}^{(n+1)} = \underset{\hat{\mathbf{p}}^{(\mathbf{n})}}{\operatorname{argmin}} \left\| \mathbf{E}^{(n)} - (\mathbf{A}(\hat{\mathbf{p}}^{(n)}) + \hat{\Gamma}) \right\|_F^2 + \lambda \left\| \hat{\mathbf{T}}^{(n)} \right\|_F^2 \tag{3.9}$$

With the error being

$$\mathbf{E}^{(n)} = \tilde{\mathbf{T}}_3^{(n)} - \mathbf{T} \tag{3.10}$$

and $\tilde{\mathbf{T}}_3^{(n)}$ being the rank-3 approximation as calculated with 3.8.
If the regularization term lambda is set to zero, the error between the low-rank approximation $\tilde{\mathbf{T}}_3^{(n)}$ and $\mathbf{T}$ is minimized. First the case of $\lambda = 0$ and all internal delays are equal is considered ($\Gamma = 0$), the Frobenius norm of $\|\mathbf{E}^{(n)} - (\mathbf{A}(\hat{\mathbf{p}}^{(n)}))\|_F^2$ in (3.9) should be minimized. This is the case when

$$\mathbf{E}^{(n)} = \mathbf{A}(\hat{\mathbf{p}}^{(n)})$$

and thus

$$C(\mathbf{E}^{(n)}) = \mathbf{W}\hat{\mathbf{p}}$$

resulting in

$$\hat{\mathbf{p}} = \mathbf{W}^{-1}C(\mathbf{E}^{(n)}) \tag{3.11}$$

in which $C$ is again the C-operation as discussed earlier in section 3.4. To avoid problems due to rank deficiency, the inverse of $\mathbf{W}$ is calculated making use of the pseudo-inverse.

### 3.5.1. Algorithm

The whole procedure of finding the best $\hat{\mathbf{p}}$ described above can be summarized in the following algorithm. The vector with unknowns $\hat{\mathbf{p}}$ is first initialized, where after a loop begins with where first the value of $\hat{\mathbf{T}}^{(n)}$ is calculated (3.12).

$$\hat{\mathbf{T}} = \mathbf{T} + \mathbf{A}(\hat{\mathbf{p}}^{(n)}) \tag{3.12}$$

Then the best rank-3 approximation of $\hat{\mathbf{T}}^{(n)}$ is computed followed by the error in (3.13) and (3.14) respectively.

$$\hat{\mathbf{T}}^{(n)} \rightarrow \tilde{\mathbf{T}}_3^{(n)} = \mathbf{U}\tilde{\Sigma}_3\mathbf{V}^T \tag{3.13}$$

$$\mathbf{E}^{(n)} = \tilde{\mathbf{T}}_3^{(n)} - \mathbf{T} \tag{3.14}$$

With the error known, the estimate of $\mathbf{p}$ can be updated according to the constraint in (3.15), which yields that the next $\hat{\mathbf{p}}^{(n+1)}$ is computed as in (3.16).

$$\hat{\mathbf{p}}^{(n+1)} = \underset{\hat{\mathbf{p}}^{(n)}}{\operatorname{argmin}} \left\| \mathbf{E}^{(n)} - (\mathbf{A}(\hat{\mathbf{p}}^{(n)})) \right\|_F^2 \tag{3.15}$$

$$\mathbf{A}(\hat{\mathbf{p}})^{(n+1)} = C_{M\times N}^{-1}\mathbf{W}\hat{\mathbf{p}}^{(n+1)} \tag{3.16}$$

Finally the Frobenius norm is computed according to (3.17) to see how well the solution with the current $\hat{\mathbf{p}}$ is.

$$F^{(n)} = \left\| \mathbf{E}^{(n)} - (\mathbf{A}(\hat{\mathbf{p}}^{(n+1)})) \right\|_F \tag{3.17}$$

If there is no more change in the Frobenius norm of (3.17), it is considered that the algorithm has converged. Otherwise, the algorithm will loop back to (3.12) to find a better estimation of $\mathbf{p}$.

## 3.6. Prerequisite conditions

From (3.1) it can be seen that every microphone and acoustic event introduce 4 unknowns, of which three are spatial coordinates for the source or receiver and 1 extra unknown for the onset time or internal delay respectively. Since the localization is rotational and transnational invariant a constraint can be placed on the coordinates of the initial object (source or receiver) can be constraint to the origin, furthermore two coordinates of the second object and 1 coordinate of the third object can be used as a reference. Resulting in a decrease of 6 unknowns, lastly $\tau_1$ is set to be 0 causing a total reduction of 7 unknowns. Resulting in the following inequality: $U^2 - 8U + 7 \geq 0$, with $U$ as the number of units, which contain a microphone as well as a speaker. This inequality can be rewritten as in (3.18).

$$U \geq (U-1)(U-7) \tag{3.18}$$

Thus, there are at least seven units required for the algorithm to work.

## 3.7. The regularization term lambda

Due to the extensive way of iterating of the algorithm to find the most optimal solution of $\hat{\mathbf{p}}$, this can take a long time since the convergence rate can be very slow [12]. This can be, given the time constraints in the Program of Requirements (Chapter 2) a serious problem, especially when the number of units in the network is large. Calculating the SVD and the pseudo-inverse in (3.11) take relatively seen a lot of computation power every iteration (A.1.4). If the number of iterations can be brought back by converging faster to a solution, this can save computation power and thereby time.

To speed up the converging process, an additional constraint is added when determining the next value of $\hat{\mathbf{p}}$ to minimize $\lambda\|\hat{\mathbf{T}}^{(n)}\|_F$ (3.9). Here, lambda is a factor to determine how dominant the effect of this additional constraint is. Due to this, the change in $\hat{\mathbf{p}}$ from one iteration to the next is larger. However, at a certain moment this change is too large to convert to an optimal solution. Therefore, $\|\hat{\mathbf{T}}^{(n)}\|$ should be monitored and when the change between two iterations is less than a certain threshold lambda must be set to zero to converge fully. In order to take this extra constraint into account, the computation of the next estimation of $\mathbf{p}^{(n)}$ has to be changed, starting with the part that has to be minimized in (3.19).

$$\left\| \mathbf{E}^{(n)} - \mathbf{A}(\hat{\mathbf{p}}^{(n)}) \right\|_F^2 + \lambda \left\| \hat{\mathbf{T}}^{(n)} \right\|_F^2 \tag{3.19}$$

Expanding $\hat{\mathbf{T}}^{(n)}$ results in

$$\left\|\mathbf{E}^{(n)} - \mathbf{A}(\hat{\mathbf{p}}^{(n)})\right\|_F^2 + \lambda \left\|\mathbf{T} + \mathbf{A}(\hat{\mathbf{p}}^{(n)})\right\|_F^2$$

When minimizing the right Frobenius norm, note that $\mathbf{T}$ is a constant matrix and is therefore ignored

$$\left\|\mathbf{E}^{(n)} - \mathbf{A}(\hat{\mathbf{p}}^{(n)})\right\|_F^2 + \lambda \left\|\mathbf{A}(\hat{\mathbf{p}}^{(n)})\right\|_F^2$$

Changing the structure with the C-operation gives

$$\left\|C(\mathbf{E}^{(n)}) - \mathbf{W}\hat{\mathbf{p}}^{(n)}\right\|_F^2 + \lambda \left\|\mathbf{W}\hat{\mathbf{p}}^{(n)}\right\|_F^2 \tag{3.20}$$

Equation 3.20 can be recognized as a Regularized Least Squares (RLS) problem. Such a RLS problem has an equation of the following form

$$\left\|\mathbf{A}\mathbf{x} - \mathbf{y}\right\|^2 + \lambda \left\|\mathbf{x}\right\|^2 \tag{3.21}$$

an solution for $x$ of the form (A.2)

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{y} \tag{3.22}$$

where $\mathbf{I}$ is the identity matrix and $\lambda$ is a constant. When furthermore making use of the fact that $\|\mathbf{w}\| \leq \|\mathbf{X}\mathbf{w}\|$, (3.20) can be solved for $\hat{\mathbf{p}}^{(n)}$

$$\hat{\mathbf{p}}^{(n)} = (\mathbf{W}^T\mathbf{W} + \lambda\mathbf{I})^{-1}\mathbf{W}^T C(\mathbf{E}^{(n)}) \tag{3.23}$$

# 4

# Implementation

This chapter will describe the implementation of the studied algorithm. After that the results of the various simulations will be shown and discussed.

## 4.1. MATLAB implementation

The implementation of the algorithm is done in MATLAB, since it is an environment in which it is easy to test the behaviour of the algorithm. The algorithm that is being implemented is the one described in subsection 3.5.1. This function of the algorithm takes as argument the measured time of arrivals and returns the corrected relative time of arrivals. For testing purposes, there are among others another few input arguments required. All arguments and returned values by the function are listed in Table 4.1, all variables were the type is not underlined are arguments for testing only.

Table 4.1: Input and output variables of the function.

| **Type** | **Name** | **Dimensions** | **Description** |
|---|---|---|---|
| Input | t | $(I \times J)$ matrix | Measured TOAs |
| Input | real_p | $(I+J-1) \times 1$ vector | Real values in $\mathbf{p}$ |
| Input | mnoi | Value | Maximum number of iterations |
| Input | threshold | Value | When the change in Frobenius norm is below this value, the solution is considered converged |
| Input | lambda | Value | The value of lambda |
| Input | threshold_lambda | Value | Determines under which value to set $\lambda = 0$ |
| Output | T_hat | $(I-1 \times J-1)$ matrix | The corrected relative TOAs |
| Output | p | $(I+J-1) \times 1$ vector | The estimated unknowns in $\mathbf{p}$ |
| Output | history | $(n \times 2)$ matrix | Stores the value of the Frobenius norm and the difference in $\mathbf{p}$ each iteration |

The complete implementation of the algorithm, including the C-operation and the inverse C-operation discussed in section 3.4 can be seen in A.1. Note that all the arguments and return values for testing are left out.

For the first step in the algorithm, the initialization of $\hat{\mathbf{p}}$, the unknowns are all initialized at random values uniformly distributed over the range specified in the Program of Requirements (2). This is for the internal delays between 0 and 20ms and for the onset times between 0 and 30ms.

With respect to the constants in the algorithm, the threshold on when to determine when a solution is converged is set to $10^{-12}$. When the change in the Frobenius norm of (3.17) between two iterations is below this threshold, the searching process of $\mathbf{p}$ is aborted. With this threshold and when the algorithm converged to the right solution, the searching process is aborted due to this threshold when the error of $\hat{\mathbf{p}}$ is in the order of tens of nanoseconds or lower. This results in an error in distance far less than a millimeter. However, this threshold can be lowered to for example $10^{15}$. Despite this can in some cases result in a way better estimation of $\mathbf{p}$ (up to picoseconds or less), this takes a lot of computation time. This is an even larger problem when $\hat{\mathbf{p}}$ is not converging to the optimal solution.

The maximum number of iterations for the program is set to $10^6$. This number is only reached if a solution does not converge to one that triggers the threshold described previously. This value is chosen well above the average worst result, to make sure that this will not stop a slowly converging process. The values of lambda and the corresponding threshold will be determined later on.

## 4.2. Simulation

In order to test the algorithm a simulation is used. The input of the algorithm is a matrix $t$ of size $I \times J$, where $I$ is the number of receivers and $J$ is the number of speakers or acoustic events as can be seen in Table 4.1. First, the random locations of all the units are set. The locations are uniform distributed in an area of predetermined sizes $x_{max}$, $y_{max}$, $z_{max}$, which is done by the `random`-function included in MATLAB. To calculate the received times in $t$ the model of (3.1) is used, which is also shown below.

$$t_{ij} = \frac{\|\mathbf{r}_i - \mathbf{s}_j\|}{c} + \tau_j + \delta_i + v_{ij}$$

In the simulation the measurement noise $v_{ij}$ is ignored and all internal delays are the same ($\delta_i = \delta_1$). In this way, the assumptions made when building the algorithm are completely valid and is it possible to see if the algorithm works well under these circumstances. Due to this, the final algorithm is a simplified version of the reality. To create a better approximation of reality a non linear least squares approach should be taken. The values of the internal delays $\delta$ are chosen between 0 and 20ms, and the values of the onset times are chosen in the range of 0 to 30ms. Furthermore, the speed of sound (in air) $c$ is set to 343m/s.

## 4.3. Results

### Accuracy

In the realistic scenario, the units of Bosch are used. This results in a network with the same number of receivers as the number of senders. To simulate the performance, a test is done with different number of units in the range of 12 up to 32 with incremental steps of 4 units. For each number of units, 20 tests are done and in every test, the locations of the units were again random chosen in an area with dimensions of $30 \times 30 \times 12\text{m}^3$. The results of the corresponding final value of the Frobenius norm is shown in Figure 4.1. In this figure the vertical bar indicates the range of the results and the line is crossing at the average error.
In Figure 4.1, a strange behaviour can be seen in test of 16 units. The error of the regular test results is quite significant, this is due to one or a few outliers which effect the average heavily. To cope with these errors the norm 3.17 is evaluated before it is accepted as being a good solution. If the norm is below $8 \times 10^{-7}$ the solution is accepted while the norm is discarded when this is not the case. This results in the filtered test results. Which has an acceptable error margin.
The result of this filtering procedure is shown in the same plot, and one can immediately notice that the filtering of the false estimation was done well. Furthermore, the Frobenius norms were all in the range between the $10^{-8}$ and $10^{-9}$.
When these solutions are converted to an error on the estimated internal delays and onset times, the error can be seen in Figure 4.2. The error is calculated when the algorithm was converged as the difference between the estimated value $\hat{\mathbf{p}}$ and the real value $\mathbf{p}_{real}$ according to (4.1).

$$E_p = \left\| \hat{\mathbf{p}} - \mathbf{p}_{real} \right\| \tag{4.1}$$

The norms of the errors are in the order of around $10^{-7}$. The corresponding errors in time are all less than a microsecond and with a higher number of units in the network this error is most of the time even lower than 100 nanoseconds. The resulting errors in distance are now in the worst case a millimeter.

The same test is also done when the units are situated in a smaller area, namely one with dimensions of $12 \times 12 \times 2\text{m}^3$. The result of the Frobenius norms (left) and the results of the corresponding error in the estimation of delays (right) are shown in Figure 4.3, both with and without filtering out the results ending with a final Frobenius norm considered too high as discussed earlier. As can be seen the error in the Frobenius norm made in the estimation are on average between $10^{-8}$ and by increasing the number of units down to $10^{-9}$. Converting this into physical distances, the resulting error is on average below a centimeter. With a higher number of units this goes down to tenths of millimeters on average and still less than a millimeter at a maximum.

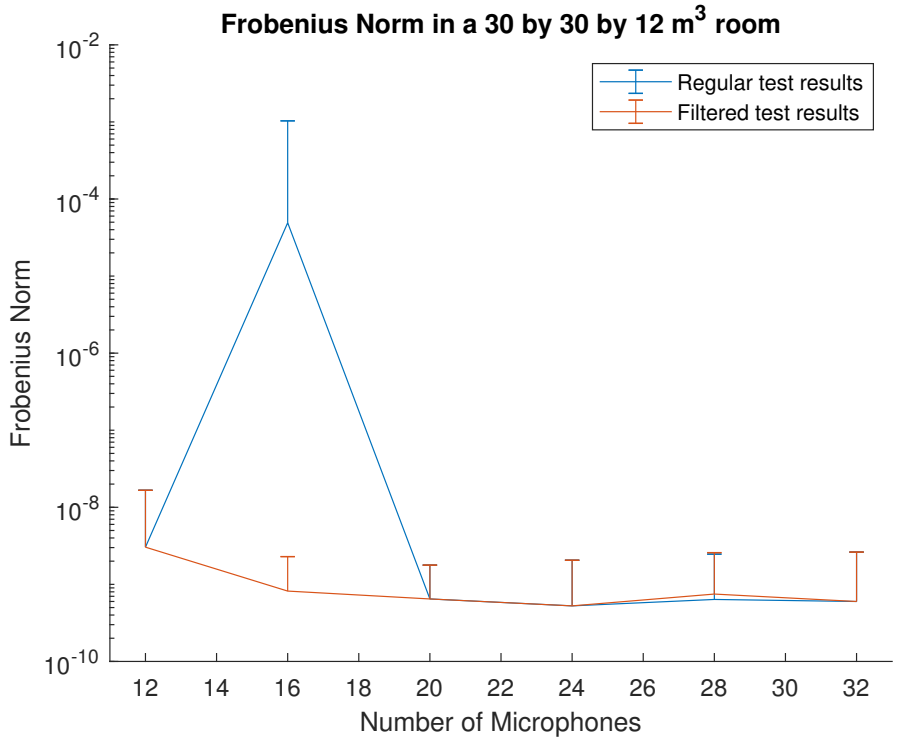


Figure 4.1: Frobenius norm in an area of $30 \times 30 \times 12\text{m}^3$ for a different number of units.

## Computation time

It is also important that the execution time of the algorithm is withing reasonable limits. This test is done in two different setups. In the first simulation a setup in a small area is used with dimensions of $12 \times 12 \times 2\text{m}^3$ with 20 units. The second simulation is situated in a large area with dimensions of $30 \times 30 \times 12\text{m}^3$ in which 120 units are placed. The tests are repeated 25 respectively 15 times and every run the units are placed at different random positions and have different internal delays and onset times. The results of these tests are shown in a box plot in Figure 4.4 and as done in the same way as before, results considered false are not taken into account. More details on the timing in A.1.4.
Taking a look at the results it can be seen that in case of the small area with 20 units the average execution time is just above 6.6 seconds and most of the runs took less than fifteen seconds. The maximum measured execution time was 35.5 seconds. When the number of units is increased to 120 in the large area, the execution time is on average 84.2 seconds and the maximum time a single run took was 105.4 seconds.

## Regularization term lambda

In order to test the performance and find an ideal value of the regularization term lambda, multiple tests were done. There was a set of test done in a room where the locations of the units as well as the delays were kept the same. Also was the same random initialization of $\hat{\mathbf{p}}$ used to compare the performance of the different values of lambda.
However, when looking at the results of the tests corresponding to the different values of lambda it was noticed that there was a lot of variance for which lambda the algorithm converged the fastest. Given the results of the simulation, this does not only depend on the spatial locations of the units, but also on the to be estimated delays of the units. As can be seen in Figure 4.5, there is a difference in which lambda performs best, but the variance is that big, that an optimal value of lambda for all situations could not be determined.

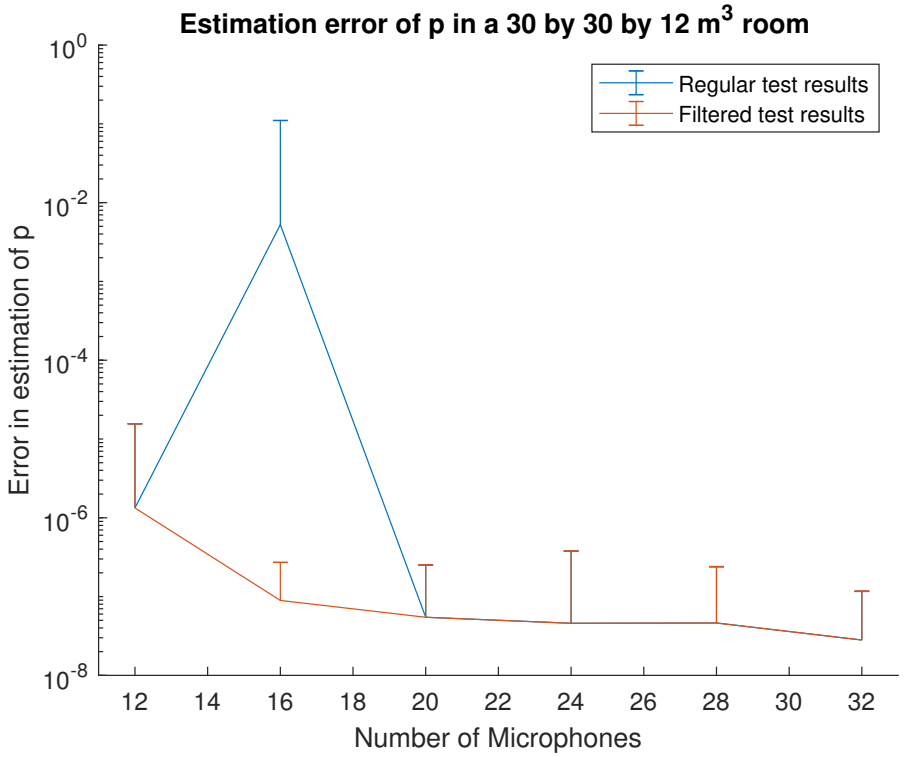


Figure 4.2: The norm as in (4.1) in a test in an area of $30 \times 30 \times 12\text{m}^3$ for a different number of units.

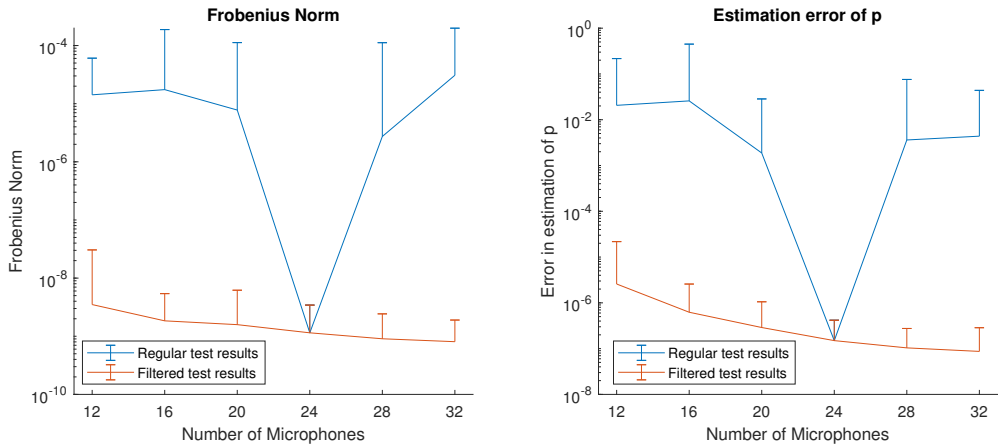


Figure 4.3: The results of the tests in an area of $12 \times 12 \times 2\text{m}^3$ for a different number of units

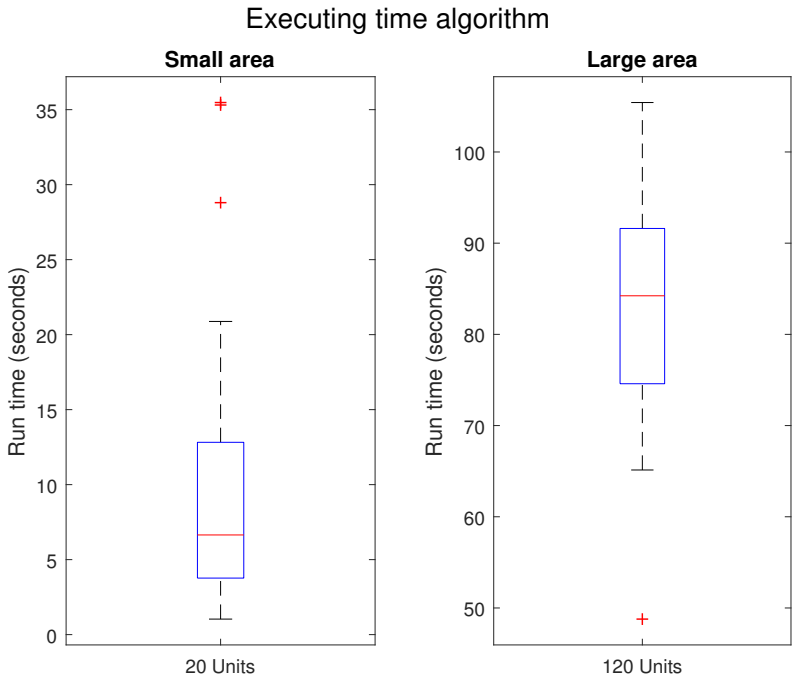


Figure 4.4: The execution speed of estimating the delays in two different situations

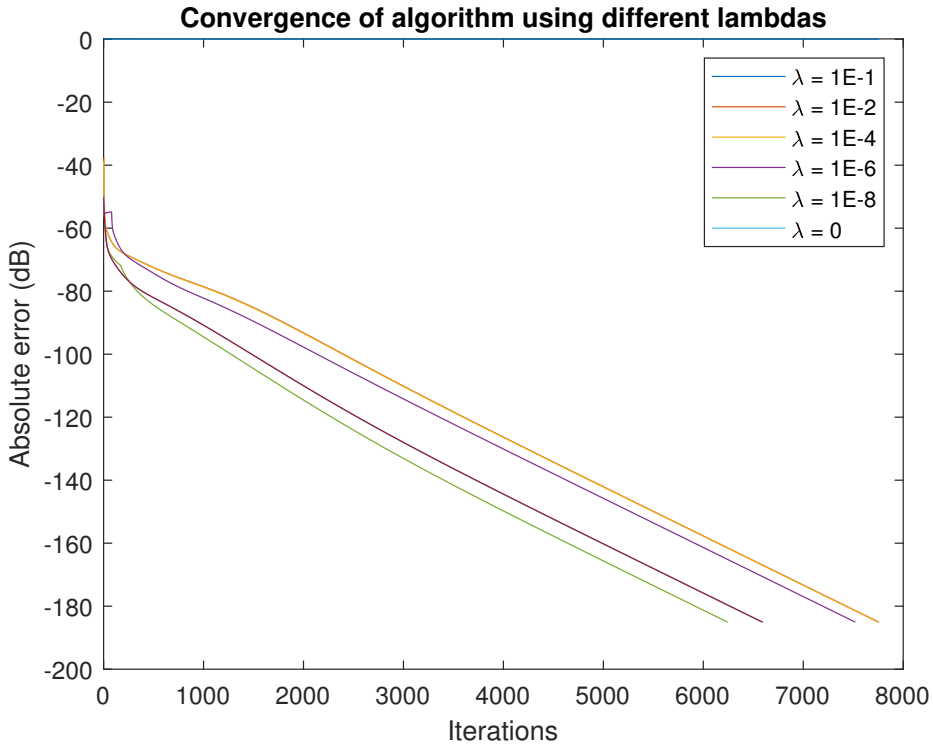


Figure 4.5: One run with the same initialization for different values of parameter $\lambda$.

# 5

# Discussion

The results of the implemented algorithm with the error correction are promising. This due to the level of the error in the estimation of $\mathbf{p}$. This error is in in the order of $10^{-6}$s which would result in an error estimation in the order of $10^{-4}$m. Since this complies with the requirement of a maximum error of 2 cm that was established in the program of requirements (Chapter 2), the algorithm is sufficiently accurate. The required scalability of the algorithm was also sufficient since the test would converge in within two minutes when simulating with all 120 units and less than one minute was required for the smaller setup of 20 units. This is within the allowed time-frame that was outlined in the program of requirements.

Although the algorithm sufficed for the stated requirements the regularization term lambda was introduced to increase the rate of convergence. The dimension of one specific room was taken in order to find the optimal lambda for one specific setup. This due to the fact that the effect of lambda is was different in each simulation setup. With the iterative approach that was taken the optimal solution could be difficult to find since there were many possibilities for the correct value. In the end the results of the simulations with different values for lambda were too inconclusive to be able to find an optimal lambda which would consistently provide the fastest rate of convergence for the simulated setup.

Due to the initial difficulties in the project, which partly resulted from the uncertainty of the accessibility of the software stack from Bosch there was no time to implement the algorithm with the real test setup. Therefor there are no results of that setup.

# 6 Conclusion

## 6.1. Conclusion

The original goal of this thesis was to improve the results of the measurements done by the ranging group, so the estimation of the locations of the localization group are more accurate (Figure 1.3). The initial aim to improve that was by synchronizing the clocks between all units (nodes) in the network, but when it became clear that there was already a good synchronization implemented on the system kept in mind for the project, the focus shifted. This is due to the improvement that is possible when trying to make a better synchronization compared to the improvement that can be achieved in estimating onset times and internal delays. There was more room for improvement since it was initially unaccounted for and therefore it became the main focus of this thesis.

The implemented algorithm for estimating the onset times and internal delays is able to estimate them to an accuracy of about a microsecond or better. This is under the assumptions given in Chapter 2 that the estimated delays are in the order of few tenths of milliseconds. In terms of distance, the error can be brought back from an error of about ten to twenty centimeters to an error of less than a millimeter.

Looking back to the Program of Requirements, with the focus on the requirements set for the subgroup of estimating the onset times and internal delays, the following conclusions can be drawn. With respect to the accuracy the maximum permitted error which is about 58$\mu$s, the achieved accuracy is far better.

Taking a look at the scalability of the proposed solution and the corresponding timing of the algorithm, it can be concluded that the algorithm still works well, even for a large number of units in the network. With an average and worst duration of 7 and 36 seconds respectively for twenty units, the requirement of less than 60 seconds is met. Also when the number of units is increased to 120, (which is the maximum number of units in the system) the average converging duration is 85 seconds, with a worst case duration 106 seconds.

In the Program of Requirements there is no requirement that specified a minimum number of units the algorithm should work with, only a maximum. Despite this, it is important to note that the algorithm is not intended to work well with only twelve nodes or less. Given the initial problem that has to be solved, the proposed solution is still very useful.
The attempt in trying to let the algorithm converge faster to a good estimate of the unknown delays did not lead to a better overall performance. This is due to the fact that the ideal value of lambda seems to depend on the positioning of the microphones, with as a result that it varied which value of lambda was best. One could of course try to optimize the lambda in a specific situation, but this is not useful when the application is kept in mind. Trying to figure out which value of lambda is the fastest costs far more computation time compared to executing the algorithm one time with a pre-set lambda to estimate the delays. All in all incorporating lambda into the algorithm currently does not result in the desired effect. This is not a problem according to the program of requirements (Chapter 2), since the restrictions on time were already met without the regularization factor.

## 6.2. Future work

Despite of the algorithm performing accurately enough (with respect to the constraints of Chapter 2) in our simulations, there are possibilities to improve the algorithm. Starting with the observation that the algorithm can converge to a local rather than a global minimum which results in a wrong estimation of the delays. To filter out these error, a simple check is performed at the last step the Frobenius norm (3.17). If this result is above a specific threshold the estimation is considered false. However, to improve the whole process of estimating the delays some modifications can be made while iterating. As described in section 3.5, when the change in Frobenius norm is below a predetermined threshold, the estimation is considered converged and the iterating is stopped. If another check on the same change in Frobenius norm with a higher threshold is added, the algorithm can determine if it is almost converged. When this is the case, the value of the Frobenius norm can be evaluated and if this is below a certain threshold, the algorithm is probably converging to a good estimation of the delays. If the Frobenius norm is above this specified threshold, the estimation of the algorithm will end up in a local minimum. Thus the algorithm should stop iterating and restart itself with a new random initialization in order to be able to converge to the correct solution.

Another possibility to optimize the estimation is to determine a value of lambda that works best in most scenarios. This can be done by doing simulations with different delays and different locations of the units. After evaluating all the data a decision can be made on which value of lambda works best in most (practical) circumstances.

In the design it is stated that all internal delays were considered equal. How valid this assumption is depends on the hardware used in a practical scenario. If not, the solution should be found through a non linear least squares approach.
To overcome other deficiencies of the algorithm a follow up algorithm could be implemented. This is the algorithm used in [13].

# A

# Appendix

## A.1. Algorithm in MATLAB

### A.1.1. C-operation

Below can be seen how the C-operation which is discussed in section 3.4 is implemented in MATLAB. This operation is required for the algorithm to function properly.

```
function [ Vector ] = Cop( Matrix )
%C operator converts a matrix into a column vector
%   as described in the paper "Auto-localization in ad-hoc microphone arrays"
    Vector = Matrix(:);
end
```

### A.1.2. Inverse C-operation

Below can be seen how the inverse C-operation discussed in section 3.4 is implemented in MATLAB.

```
function [ Matrix ] = iCop( Vector, M, N )
%inverse C operator converts a column vector into a MxN matrix
%   as described in the paper "Auto-localization in ad-hoc microphone arrays"
    if N*M ~= length(Vector)
        display(['WARNING: can not fit a vector with ', num2str(length(Vector)), ' items into a ', num2str(M), 'x',
            ↪ num2str(N), ' matrix with ', num2str(N*M), ' items']);
    end;
    Matrix = reshape(Vector,[M,N]);
end
```

The main algorithm is shown on the next page.

### A.1.3. Algorithm for estimating delays

Below is the implemented algorithm for determining the internal delays and onset times written in the MATLAB programming language. This is described in section 3.5.1.

```
function [ T_hat ] = HeuKleGau( t )
%Internal delay and onset time estimation
%   First part (algorithm 1) of the paper by Heusdens, Kleijn and Gaubitch
%   "auto-localization in ad-hoc microphone arrays"

mnoi = 1E6; % maximum number of iterations
threshold = 1E-12; % threshold if converged
threshold_lambda = 1E-8; % threshold when lambda goes to 0
lambda = 1E-2;

I = size(t,1); % number of microphones
J = size(t,2); % number of speakers
T = zeros(I-1,J-1);
for i = 2:I
    for j = 2:J
        T(i-1,j-1) = t(i,j)^2 - t(1,j)^2 - t(i,1)^2 + t(1,1)^2;
    end;
end;
W = zeros((I-1)*(J-1), I+J-1);
row = 0;
for j = 2:J
    for i = 2:I
        row = row + 1;
        W(row, 1) = 2*(t(1,j) - t(1,1));
        W(row, i) = -2*(t(i,j) - t(i,1));
        W(row, I+j-1) = -2*(t(i,j) - t(1,j));
    end;
end;

F_previous = 0;

% 0: initialize p(n=0)
p = [random('unif', 0, 2E-2, number_of_receivers, 1); random('unif', 0, 3E-2, number_of_senders-1, 1)];

% While not converged
for n=1:mnoi
    % 1: Compute T_hat(n) = T + A(p(n))
    T_hat = T + iCop(W*p, I-1, J-1);

    % 2: Rank-3 approximation of T_hat(n): T_wave3(n) = US3V'
    [U, S, V] = svd(T_hat);
    T_wave3 = U(:,1:3)*S(1:3,1:3)*(V(:,1:3)');

    % 3: Compute error: E(n) = T_wave3(n) - T
    E = T_wave3 - T;

    % 4: Update estimate of p
    p = (W'*W + lambda*(I+J-1)*eye(I+J-1))\(W'*Cop(E));

    % 5: Update estimate of A(p): p(n+1) = iCop(W*p(n+1),I-1,J-1)
    A = iCop(W*p, I-1, J-1);

    if lambda == 0
        % 6: Compute Frobenius norm: F(n) = || E(n) - A(p(n+1)) ||_F
        F_new = norm(E - A, 'fro');

        % 7: Exit if converged
        if abs(F_previous - F_new) < threshold
            break;
        end;
    else
        % 6: Compute Frobenius norm: F(n) = || T_hat(n) ||_F
        F_new = norm(T + A, 'fro');

        % 7: Set lambda to zero
        if (abs(F_previous - F_new) < threshold_lambda)
            lambda=0;
        end;
    end;
    F_previous = F_new;
end;
end
```

### A.1.4. Measurements of execution time

With respect to the performance of the written algorithm in MATLAB, the execution time of the algorithm is measured. This is done using the build-in timer in MATLAB. During the testing of the execution time in

MATLAB a computer available at the TU Delft is used, on which a 64-bit version of Windows 10 Enterprise (build 14393.2312) ran with MATLAB 2016b. The computer had 8GB of RAM memory and a Intel Core i5 4690 processor with a clock speed of 3.50GHz.

### Relative computation time

After executing multiple test, the relative time of different function calls are measured with the MATLAB Profiler. The steps in the algorithm with the highest computation time are listed in Table A.1 below. In the algorithm no regularization term is used.

Table A.1: Relative computing times in MATLAB

| Instruction | Calls | Total Time | Relative Time |
|---|---|---|---|
| Computing SVD of T_hat | 705902 | 71.506s | 50.8% |
| Computing next p with inverse | 705902 | 46.514s | 33.1% |
| Calculating A with inverse C operation | 705902 | 9.472s | 6.7% |
| Computing T_wave3 by truncation | 705902 | 7.879s | 5.6% |
| Other | | 5.276s | 3.7% |

## A.2. Regularized Least Squares

In the case of a linear kernel in section 3.7 with $\mathbf{x}$ as unknown vector:

$$L = \min_{\mathbf{x}} \frac{1}{2}\|\mathbf{Y} - \mathbf{A}\mathbf{x}\|^2 + \frac{\lambda}{2}\|\mathbf{x}\|^2 \tag{A.1}$$

Taking the derivative with respect to $\mathbf{x}$:

$$\frac{dL}{d\mathbf{x}} = \mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{A}^T\mathbf{Y} + \lambda\mathbf{x} \tag{A.2}$$

Setting to zero:

$$\mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{A}^T\mathbf{Y} + \lambda\mathbf{x} = 0 \tag{A.3}$$

Rewriting with $\mathbf{x}$ on the left side of the equal sign:

$$\mathbf{x}(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}) = \mathbf{A}^T\mathbf{Y} \tag{A.4}$$

In which $\mathbf{I}$ is the identity matrix. Solving for $\mathbf{x}$:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{Y} \tag{A.5}$$

Which is the solution to the Regularized Least Squares problem [17].

# Bibliography


[1] R. C. Shit, S. Sharma, D. Puthal, and A. Y. Zomaya, "Location of things (lot): A review and taxonomy of sensors localization in iot infrastructure," *IEEE Communications Surveys & Tutorials*, 2018.

[2] A. Mackey and P. Spachos, "Performance evaluation of beacons for indoor localization in smart buildings," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 823–827, Nov 2017.

[3] S. P. Chepuri, R. T. Rajan, G. Leus, and A. J. van der Veen, "Joint clock synchronization and ranging: Asymmetrical time-stamping and passive listening," *IEEE Signal Processing Letters*, vol. 20, pp. 51–54, Jan 2013.

[4] L. Yi, L. Tao, and S. Jun, "Rssi localization method for mine underground based on rssi hybrid filtering algorithm," in *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, pp. 327–332, May 2017.

[5] C. Park, H. Cho, D. Park, Y. Lee, S. Cho, and J. Park, "Aoa localization system design and implementation based on zigbee for applying greenhouse," in *2010 5th International Conference on Embedded and Multimedia Computing*, pp. 1–4, Aug 2010.

[6] I. Guvenc and C. C. Chong, "A survey on toa based wireless localization and nlos mitigation techniques," *IEEE Communications Surveys Tutorials*, vol. 11, pp. 107–124, rd 2009.

[7] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 22, pp. 54–69, July 2005.

[8] J. M. Valin, F. Michaud, J. Rouat, and D. Letourneau, "Robust sound source localization using a microphone array on a mobile robot," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 2, pp. 1228–1233 vol.2, Oct 2003.

[9] T. K. Le, N. Ono, T. Nowakowski, L. Daudet, and J. D. Rosny, "Experimental validation of toa-based methods for microphones array positions calibration," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3216–3220, March 2016.

[10] A. Brutti, M. Omologo, and P. Svaizer, "Comparison between different sound source localization techniques based on a real data collection," in *2008 Hands-Free Speech Communication and Microphone Arrays*, pp. 69–72, May 2008.

[11] "Toa-based distributed localisation with unknown internal delays and clock frequency offsets in wireless sensor networks, author=Yu, Kegen and Guo, Y Jay and Hedley, Mark, journal=IET Signal Processing, volume=3, number=2, pages=106–118, year=2009, publisher=IET,"

[12] N. D. Gaubitch, W. B. Kleijn, and R. Heusdens, "Auto-localization in ad-hoc microphone arrays," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 106–110, May 2013.

[13] R. Heusdens and N. Gaubitch, "Time-delay estimation for toa-based localization of multiple sensors," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 609–613, May 2014.

[14] G. Golub and C. van Loan, "An analysis of the total least squares problem," *SIAM Journal on Numerical Analysis*, vol. 17, no. 6, pp. 883–893, 1980.

[15] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, pp. 211–218, Sep 1936.

[16] D. Poole, *Linear Algebra: A Modern Introduction*. Cengage Learning, 3rd ed., 2011.

[17] R. Rifkin and R. Lippert, "Notes on regularized least squares," May 2007.