Implementation Of A Sourcetracking System Using Cross Correlation Of Speech Signals

S.A. Atmadja

M.Sc. Thesis

Supervisors: prof. dr. ir. A. Gisolf
dr. ir. D. de Vries

Laboratory of Acoustic Imaging and Sound Control
Department of Applied Physics
Faculty of Applied Sciences
Delft University of Technology

Delft, September 2002
Graduation Committee:  prof. dr. ir. A. Gisolf
Delft University of Technology
Department of Applied Physics
Laboratory of Acoustic Imaging and Sound Control

dr. ir. D. de Vries
Delft University of Technology
Department of Applied Physics
Laboratory of Acoustic Imaging and Sound Control

prof. dr. ir. P.M.J. van den Hof
Delft University of Technology
Department of Applied Physics
Signals, Systems and Control Group

ir. J. Baan
TNO-TPD

ir. R.A. Metkemeijer
Adviesbureau Peutz & Associés
Zoetermeer

C.A.M. Wagenaar
BV Oorzaak
Amersfoort
To my father
Sound amplification in theaters generally uses a stereo pair or cluster of speakers for reproduction in the audience area. One effect occurring with this setup is that the amplified sound seems to originate from the speaker position, instead of the person speaking or singing. This mis-localization problem can be solved using a wave field synthesis system instead of a stereo configuration. This system is capable of generating a sound field that seems to originate from the actual position of the sound source. To accomplish this feat the system takes as its input the sound recorded close to the source (using a wireless microphone system), and the position of the source. The source position has to be detected using a source tracking system. In this thesis a source tracking system is proposed that records the sound signals from the source, and also from a number of microphones positioned along the edges of the theater stage. By detecting the distances between the source microphone (also called the close microphone) and the microphones at the edges of the stage (the remote microphones) the source position can be determined by solving a simple geometrical problem.

To detect the distances the signals from the close and remote microphones are divided into time windows. Then, for each remote microphone time window the cross correlation with the close microphone time window is calculated. After some additional filtering, this will result in the impulse response from the close microphone position to the remote microphone position. The impulse response will contain a large peak, called the direct sound source peak (DSS peak). This peak corresponds to the sound waves that travel directly from the source to the remote microphone position, without being reflected off walls and other surfaces. By determining the position of the DSS peak in the impulse response the time delay $\tau$ the direct sound is calculated. When this time delay is known, the distance can easily be calculated using the speed of sound.
Problems start arising, however, when the source moves across stage. As now the position must be updated frequently enough in real time so that the audience still perceives the sound as coming from the source. This necessitates the optimization of the source tracking algorithm as was originally proposed by Reeuwijk [1], on which this system is based. For example, the number of remote microphones was increased to improve tracking performance. Also the concept of time window overlapping was introduced to enable using long time windows while still maintaining a high update rate. This is important since the length of the time window has a large effect on the detection performance of the system.

The end result is a working prototype source tracking system. It incorporates many adaptations to improve the detection performance. The system is capable of tracking a moving source on stage in real time.
Samenvatting

Geluidsversterking op theaterpodia wordt normaal gesproken gerealiseerd door middel van een paar stereoluidsprekers om het geluid weer te geven in de zaal. Een fenomeen dat zich hierbij voordoet is dat het versterkte geluid lijkt te komen uit de luidsprekers in plaats van uit de persoon die spreekt of zingt. Dit probleem waarbij het geluid vanaf de verkeerde lokatie komt, kan opgelost worden door een golfveld synthese systeem te gebruiken in plaats van een stereo configuratie. Zo’n systeem kan een zodanig golfveld produceren dat het lijkt alsof het geluid afkomstig is van de werkelijke positie van de geluidbron. Om dit te realiseren heeft het systeem het directe geluid bij de bron nodig (opgenomen met een draadloos microfoon systeem) en de positie van de bron. De positie van de bron wordt gedetecteerd door middel van een bronvolgsysteem (source tracking system). In dit afstudeerverslag wordt een voorstel gedaan voor een bronvolgsysteem dat werkt door het directe geluid op te nemen, samen met het geluid dat opgenomen wordt door een aantal microfoons aan de randen van het podium. Door de afstand te bepalen tussen de close microfoon en de remote microfoons kan de positie worden berekend door een eenvoudig meetkundig probleem op te lossen.

Om deze afstanden te bepalen worden de diverse microfoonsignalen in tijdsintervalen verdeeld (time windows). Voor elk tijdsinterval wordt de kruiskorrelatie berekend tussen het close signaal en de remote signalen. Na toepassing van enige filters resulteert dit in de pulsresponsie van de bronpositie naar de remote microfoon posities. Deze pulsresponsies bevatten een grote piek die het directe geluidssignaal piek wordt genoemd (Direct Sound Source peak, DSS piek). Deze piek komt overeen met de geluidsgolven die direct vanaf de bron naar de remote microfoons propageren, zonder dat ze gereflecteerd worden door de muren of andere oppervlakken. Door de positie van deze DSS piek in de pulsresponsie te bepalen kan de tijdsvertraging van het directe
geluid worden berekend. Wanneer deze bekend is kan met behulp de snelheid van het geluid in lucht eenvoudig de afstand worden berekend.

Er doen zich echter problemen voor wanneer de bron zich over het podium beweegt. De positie moet nu vele malen per seconde worden aangepast om de illusie in stand te houden dat het geluid vanaf de bronpositie komt. Hiervoor is het nodig om het systeem te optimaliseren dat oorspronkelijk is voorgesteld door Reeuwijk [1], waarop dit systeem is gebaseerd. Het aantal microfoons, bijvoorbeeld, is vergroot om de werking te verbeteren. Tevens is het concept van overlappende tijdsintervallen geïntroduceerd om het gebruik van lange tijdsintervallen te kunnen verenigen met een hoge detectie frequentie. Dit is belangrijk omdat de lengte van de tijdsintervallen in grote mate bepalend is voor het vermogen om posities te bepalen.

Het einderesultaat is een werkend prototype bronvolgsysteem. Het bevat vele aanpassingen aan het originele systeem om de werking te verbeteren. Het systeem is in staat om een bewegende bron op het podium te volgen in ‘real time’.
## Contents

Abstract v  
Samenvatting vii  
List of Tables xiii  
List of Figures xv  

1 Introduction 1  
1.1 Direct sound enhancement 1  
1.2 Tracking of sound sources 2  
1.3 Research goals 3  
1.4 Thesis outline 3  

2 Detecting the direct sound source peak 5  
2.1 Cross correlation function 5  
2.2 Pre-whitening 8  
2.2.1 Roth processor 9  
2.2.2 Stabilization factor 10  
2.3 Peak detection 10
2.3.1 Noisy signals: Threshold value ...................................... 11
2.3.2 Coinciding reflections: Relative height ............................ 11
2.3.3 Stage size windowing .................................................... 12
2.4 Time window length ........................................................ 12

3 Determining onstage position .............................................. 15
  3.1 Introduction ................................................................. 15
  3.2 Intersecting circles ....................................................... 15
  3.3 Stage area ................................................................. 17
    3.3.1 Rejection of off-stage positions ................................ 17
  3.4 Average position ......................................................... 17
  3.5 Systematic error ......................................................... 17

4 Simulation ........................................................................... 19
  4.1 Introduction ................................................................. 19
  4.2 Remote microphone signal simulation .............................. 19
    4.2.1 Geometrical spreading ............................................. 19
    4.2.2 Travel time delay .................................................... 20
    4.2.3 Noise ................................................................. 20
  4.3 Evaluation ................................................................. 20
  4.4 Conclusions ............................................................... 22

5 Real world implementation .................................................. 23
  5.1 First prototype ........................................................... 23
    5.1.1 Microphones ......................................................... 23
    5.1.2 Digital tape recorder ............................................... 23
    5.1.3 Sound card .......................................................... 24
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.4 Software</td>
<td>25</td>
</tr>
<tr>
<td>5.2 Real time implementation</td>
<td>26</td>
</tr>
<tr>
<td>5.2.1 Hardware</td>
<td>26</td>
</tr>
<tr>
<td>5.2.2 Microphones</td>
<td>29</td>
</tr>
<tr>
<td>5.2.3 Interfacing digital audio equipment</td>
<td>29</td>
</tr>
<tr>
<td>5.2.4 Software</td>
<td>30</td>
</tr>
<tr>
<td>6 Results</td>
<td>35</td>
</tr>
<tr>
<td>6.1 First prototype results</td>
<td>35</td>
</tr>
<tr>
<td>6.1.1 Number of remote microphones</td>
<td>35</td>
</tr>
<tr>
<td>6.1.2 Stabilization factor</td>
<td>37</td>
</tr>
<tr>
<td>6.2 Real time results</td>
<td>41</td>
</tr>
<tr>
<td>6.2.1 Time Window Length</td>
<td>41</td>
</tr>
<tr>
<td>6.2.2 Time window overlap</td>
<td>42</td>
</tr>
<tr>
<td>6.2.3 Array configuration</td>
<td>43</td>
</tr>
<tr>
<td>7 Extrapolation</td>
<td>45</td>
</tr>
<tr>
<td>7.1 Introduction</td>
<td>45</td>
</tr>
<tr>
<td>7.2 Results: Detection runaway</td>
<td>46</td>
</tr>
<tr>
<td>7.3 Conclusion</td>
<td>46</td>
</tr>
<tr>
<td>8 Inversion method</td>
<td>49</td>
</tr>
<tr>
<td>8.1 Introduction</td>
<td>49</td>
</tr>
<tr>
<td>8.2 Theory</td>
<td>49</td>
</tr>
<tr>
<td>8.3 Results</td>
<td>52</td>
</tr>
<tr>
<td>9 Conclusions and suggestions for further research</td>
<td>57</td>
</tr>
<tr>
<td>9.1 Research goals</td>
<td>57</td>
</tr>
</tbody>
</table>
9.2 Comparison of conclusions ........................................ 57
  9.2.1 Time window length ........................................ 58
  9.2.2 Number of microphones .................................. 58
  9.2.3 Extrapolation ................................................ 58
9.3 Conclusions ...................................................... 59
9.4 Suggestions for further research ............................... 60

A Noise signals ...................................................... 61

B Cosine rule ......................................................... 63

Bibliography .......................................................... 65

Index ................................................................. 67
List of Tables

4.1 Initial parameter values ........................................... 22

5.1 Reference PC: Dual CPU Audio Workstation specifications ........ 27
5.2 Audio equipment ...................................................... 29
5.3 Software components ............................................... 31
List of Figures

2.1 Direct sound signal (top) and remote microphone signal (bottom). The remote signal is a slightly delayed and “polluted” version of the direct signal. ......................................................... 7

2.2 Cross correlation function of a close microphone signal with a remote microphone signal without whitening (top) and with pre-whitening (bottom). The source signal has a strong periodic component, which can be seen in the non filtered CCF. This makes it very difficult to determine a correct DSS peak. A pre-whitening filter filters out the source signal, leaving the impulse response. It reveals the true DSS peak. ......................................................... 9

3.1 Intersecting circles ................................................. 16

3.2 Systematic error due to neglected speaker height. ..................... 18

4.1 Top: Source signal (speech). Bottom: Number of valid remote microphone pairs detected. Notice that between words (where the source signal is very weak) the detection drops to zero. ......................... 21

5.1 Configuration of first prototype. ...................................... 24

5.2 The microphone array positioned along the front edge of the stage area. 25

5.3 The Tascam DA-88 digital multitrack tape recorder. .................... 26

5.4 Configuration of real time prototype. .................................. 27
5.5 The digital audio workstation. The two small boxes between the computer case and the keyboard are the IF-TAD converters. ............... 28
5.6 The TM-D4000 digital mixer, used as an A/D converter. .............. 30
5.7 Screenshot of the ‘stage area’. The red square in the middle represents the current source position on the stage area (the blue field). The red bar on the left is a meter that displays the number of detected positions averaged to derive the final position, with the full scale being 21 positions when using 7 remote microphones (see section 3.4). .......... 32
5.8 Screenshot of the settings page. Here the various parameters can be entered, like the size of the stage, the time window length, the sampling rate, etc. ................................................................. 33
6.1 Remote microphone configuration. ............................................. 35
6.2 Using 7 remote microphones. A speaker moved across stage from right to left and back, while talking into a handheld close microphone. . . 36
6.3 The same recording as in fig. 6.2, but using only microphones 1-5. Notice that the right part of the source path was not detected. ......... 37
6.4 Again as in fig. 6.2, but now with microphones 1-3. Again, the right-hand part of the path was not detected. ................................. 38
6.5 Stabilization factor \( A = 0.5 \). ....................................................... 39
6.6 Stabilization factor \( A = 0.25 \). ....................................................... 39
6.7 Stabilization factor \( A = 0.1 \). ......................................................... 40
6.8 Stabilization factor \( A = 0.05 \). ....................................................... 40
6.9 Time window overlap ................................................................. 42
6.10 The microphone array consisting of seven remote microphones. . . 43
7.1 First order extrapolation ............................................................. 47
7.2 Zeroth order extrapolation .......................................................... 47
8.1 Cross correlation functions before lining up. ................................. 50
8.2 Cross correlation functions after lining up. .............................. 51
8.3 Summing up the lined up cross correlation functions. ............... 51
8.4 Summing up peaks that do not line up, resulting in a low value for the sum. ................................................................. 52
8.5 Inversion method used an a very clear signal. Notice the intersecting circles can actually be seen in this picture ....................... 53
8.6 The CCF does not seem to have a clear DSS peak in this picture. . 53
8.7 Although the circle like features are very clear in this picture, it is not clear where the source was actually located ....................... 54
8.8 Another example that is not suitable for detecting the source position. 54
B.1 Cosine rule ................................................................. 63
Chapter 1

Introduction

1.1 Direct sound enhancement

In theaters and auditoria around the world the use of electro-acoustical systems for the purpose of speech enhancement has become commonplace. A public address system may be considered standard equipment for any venue where a speaker may address the public. The traditional configuration for such a system may consist of a stereo pair of loudspeakers, placed at either side of the stage area, amplifying what the speaker is saying into the microphone. A well known problem with this configuration is the mis-localization of the source. The listeners can see the speaker standing in the middle of the stage, but they hear the sound coming from the sides of the stage, where the loudspeakers are located. Also, when the speaker starts walking around the stage area, the amplified sound still seems to be located at the static positions at either side of the stage.

A possible solution is the use of a wave-field synthesis system developed by the Laboratory of Acoustical Imaging and Sound Control at Delft University of Technology. Using a loudspeaker array in conjunction with digital signal processors (DSP’s) it is possible to recreate the wavefield in a more natural sounding way. The wavefield synthesis system is capable of making the amplified sound appear as if it was generated at the position of the real source. The audience will hear the sound coming from the position of the speaker, resulting in a much more natural sounding listening experience.
1.2 Tracking of sound sources

To make this system work, it needs both the speech signal that needs to be amplified, and the position of the speaker, from which the amplified sound should appear to originate. When the person speaking is located at a fixed position, it is sufficient to place a microphone close to the speaker, and to program the signal processors with the coordinates of the speaker position. When the speaker starts to move around stage, as is the case with many theater performances (drama, musical, opera etc.), the coordinates are no longer known beforehand. There are two possible strategies for solving this problem: remote miking and close miking.

When using the remote miking technique, a number of directional microphones are used from a relatively large distance. They may be suspended above the stage area, or installed at the edge of the stage area. Each microphone covers a small area of the stage. All microphones combined cover the entire stage. A signal picked up by one of the microphones is assumed to originate from the position this microphone is pointed at.

This method, however, has many disadvantages. Because the stage area is subdivided into many regions, each covered by a microphone, some overlap will occur. So the speech signal is picked up by multiple microphones. And because there are only a limited number of microphones available, the number of possible source positions is limited. This may cause the amplified sound to jump around from position to position when the speaker moves across the stage.

Another disadvantage is the distance between the speaker and the microphone. As the distance is relatively large, a lot of ambient noise is also picked up. And because of the high gain needed to amplify the distant speech signal, the risk of acoustical feedback is considerable.

The alternative method uses a microphone close to the speaker. Possible in combination with a wireless system. This method is far less susceptible to acoustical feedback, as the speech signal is very strong so close to the source. The disadvantage, however, is that now the position of the speaker is unknown. An additional system must be used to determine the position of the speaker. Possible techniques that could be employed include infra-red based systems or video based systems. The drawbacks are that these systems potentially require a lot of extra equipment to be installed, making the total system very complex.
The advantage of using cross-correlation based techniques is that very little extra equipment is needed. In addition to the close microphone, only a number of remote microphones need to be installed, preferably at the edge of the stage area.

### 1.3 Research goals

The goal of this research is to implement a system capable of tracking a moving sound source on a theater stage. This system is based on the research done by Reeuwijk, and is described in his M.Sc thesis [1]. The first step in tracking a source using speech signals is to determine the travel time of the speech signal from the source to microphones placed at the edges of the stage area. The signals picked up by these microphones are then cross correlated with the source signal, which is picked up by a wireless microphone system. By determining the position of the highest peak in the cross correlated signals, the travel time is determined. The next step is to take the travel time data from the various microphones, which is easily converted into distances, and calculate a two dimensional position on the theater stage.

When the source tracking system is capable of determining the position in real time, it is possible to use this data to control a wavefield synthesis system for amplification of the speech signal. The amplified sound will appear to originate from the actual position of the speaker. When the speaker moves on stage, the amplified sound will follow the speaker.

Computer simulations have shown that this approach to source tracking does work [1]. This thesis describes the process of implementing the system, and testing it with real-world data.

### 1.4 Thesis outline

This thesis starts in chapter 2 with a theoretical examination of the principles behind the source tracking algorithm and specifically, the determination of the Direct Sound Source peak (DSS), using cross correlation. Also, the use of a pre-whitening filter is examined.

Chapter 3 follows with the methods used to determine a 2-d position, using the infor-
mation obtained in the previous chapter.

Chapter 4 shows how simulated data is used to verify the performance of the system during the initial stages of development.

Chapter 5 describes the implementation of a prototype source tracking system.

Chapter 6 discusses the results obtained using the source tracking prototype.

Chapter 7 deals with the possible use of extrapolation, in case the system is unable to detect a position due to a weak signal.

Chapter 8 describes a possible alternative method to determine the source position from cross-correlation data.

Chapter 9 presents the conclusions and contains suggestions for possible further research.
Chapter 2

Detecting the direct sound source peak

2.1 Cross correlation function

The cross correlation function (CCF) between two signals describes how much the two signals resemble (correlate) each other. The auto correlation function (ACF) is similar to the cross correlation function, except that it correlates a signal with itself. It will have a peak at $t = 0$. The ACF describes how much a signal resembles itself if it is shifted by a time lag $t$. For example: a periodic signal with period $T$ will show peaks in its ACF at $t = nT$ ($n = ..., -1, 0, 1, 2, ...$). When a signal is correlated with a copy of itself, shifted by a time lag $t$, the correlation function will show a peak at this value of $t$.

When we record a speech signal $x_1$ with a microphone positioned very close to the source (called the close microphone signal), and a delayed (and more or less reverberated) version of this signal $x_2$ at some distance away (for example at the edge of a stage), $x_2$ (called the remote microphone signal) will resemble $x_1$, except that it is shifted by a time lag $t$. This time lag equals the time the sound waves need to travel from the source to the remote microphone. Using the speed of sound\(^2\) we can calculate the distance between the two microphones if we can determine the time lag $t$.

---

\(^1\) A more thorough examination of the CCF can be found in [2] and [19]

\(^2\) The speed of sound $c$ in air at $20^\circ C$ is assumed to be $340m/s$
Chapter 2: Detecting the direct sound source peak

The signal $x_1$ picked up by the close microphone consists of the source signal $s(t)$ (a person singing or speaking), convoluted with the impulse response from the source position (the mouth of the person) to the close microphone position $p_1$.

\[ x_1(t) = s(t) \otimes p_1(t) + n_1(t) \quad (2.1) \]

The $\otimes$ symbol denotes convolution.

Also picked up is the noise signal $n_1$. Because of the fact that the close microphone is located very close to the source $p_1(t)$ can be approximated by a Dirac delta pulse $\delta(t)$, such that

\[ p_1(t) \approx \delta(t), \quad (2.2) \]

\[ x_1(t) \approx s(t) + n_1(t). \quad (2.3) \]

At the remote microphone position signal $x_2$ is picked up, which consists of the source signal $s(t)$ convoluted with the impulse response at this position $p_2(t)$. Added to this is the noise signal $n_2$, which is assumed to be uncorrelated with the noise picked up by the close microphone ($n_1$), such that

\[ x_2(t) = s(t) \otimes p_2(t) + n_2(t). \quad (2.4) \]

The subscript 2 will be left out in the following equations since it is the only impulse response playing a role.

In fact, what we are trying to do is to determine the impulse response from the source to the remote microphone position. The impulse response will have a large initial peak, which corresponds to the direct sound arriving at the remote microphone position, followed by reflections from the ceilings, floors and walls. The position of the direct sound source peak is equivalent to the time delay.

To determine the impulse response we use the correlation theorem which says that the correlation function of two real functions $g$ and $h$ is one part of the Fourier transform pair

\[ \text{Corr}(g(t), h(t)) \Longleftrightarrow G(f)^* H(f) \quad (2.5) \]

where $G$ and $H$ are the Fourier transforms of $g$ and $h$, and the asterisk denotes complex conjugation. The frequency domain side of the transform pair ($G^* H$) is also called the cross power spectrum.
The Fourier transforms of $x_1$ and $x_2$ are
\[
X_1(f) = S(f) + N_1(f) \tag{2.6}
\]
and
\[
X_2(f) = S(f)P(f) + N_2(f) \tag{2.7}
\]
respectively.

Here we used the convolution theorem, which is closely related to the correlation theorem. In short it says that convolution in the time domain is equivalent to multiplication in the frequency domain.
\[
r(t) \otimes s(t) \Leftrightarrow S(f)R(f) \tag{2.8}
\]
For now we will disregard\(^3\) the noise signals $N_1$ and $N_2$, so that we have
\[
X_1(f) = S(f) \tag{2.9}
\]
\(^3\)Appendix A will show that the noise signal can indeed be disregarded.
Chapter 2: Detecting the direct sound source peak

and

\[ X_2(f) = S(f)P(f) \]  \hspace{1cm} (2.10)

and after complex conjugation and multiplication

\[ X_1(f)^*X_2(f) = |S(f)|^2 P(f) \]  \hspace{1cm} (2.11)

What we have now is the source signal power spectrum multiplied with the Fourier transform of the impulse response. In the time domain this is equivalent to the source signal auto correlation function convoluted with the impulse response:

\[ |S(f)|^2 P(f) \leftrightarrow \text{Corr}(s(t), s(t)) \otimes p(t) \]  \hspace{1cm} (2.12)

In general, the impulse response of a hall can be modeled by a Dirac delta pulse, positioned to correspond with the Direct Sound Source (DSS) peak, followed by smaller peaks generated by reflections off walls, floors and ceilings. In this case, however, this nice sharp delta pulse is convoluted with the source signal auto correlation function. The result is that the direct sound source pulse is no longer a well defined peak, but it is 'smeared' out. The shape of the resulting peak is determined by the shape of the source signal auto correlation function. This process might be easier to understand when looked at from a frequency domain perspective. A Dirac delta pulse has a uniform (white) spectrum. This is also true in case the delta pulse is shifted in time, as a time-shift only affects the phase of the spectrum, and not the amplitude. In equation 2.11 the delta pulse response spectrum is multiplied by the source signal power spectrum, which might not have a uniform spectrum. In fact, if the source signal is a speech signal, it is highly unlikely that it has a white spectrum. This results in a cross correlation function which has a non-white spectrum, and therefore no longer resembles a Dirac delta pulse.

2.2 Pre-whitening

To compensate for the ‘non-whiteness’ of the source signal a pre-whitening filter is applied to the signal. Its function is to ‘sharpen’ the peaks (in essence making the spectrum white). This makes detecting the peaks much easier and more accurate. However, if we whiten the signal, information about the shape of the cross correlation function is lost as the CCF is reduced to the impulse response. But in this case this is not a problem, since we are only interested in the position of the peaks, and the CCF itself is irrelevant.
Chapter 2: Detecting the direct sound source peak

Figure 2.2 Cross correlation function of a close microphone signal with a remote microphone signal without whitening (top) and with pre-whitening (bottom). The source signal has a strong periodic component, which can be seen in the non filtered CCF. This makes it very difficult to determine a correct DSS peak. A pre-whitening filter filters out the source signal, leaving the impulse response. It reveals the true DSS peak.

2.2.1 Roth processor

The pre-whitening filter used here is the Roth processor. The Roth processor has the desirable effect of suppressing those frequency regions where the signal strength is low, and emphasizing the regions where the signal is strong.

The Roth filter replaces the cross correlation function with the transfer function \( H(f) \)

\[
H(f) = \frac{G_{x_1x_2}}{G_{x_1x_1}} \tag{2.13}
\]

where \( G_{ab} \) is the cross power spectrum of \( a \) and \( b \). If we insert the variables \( X_1 \) and \( X_2 \) into this equation we get, using equations 2.9 and 2.10:

\[
H(f) = \frac{X_1^*(f)X_2(f)}{X_1^*(f)X_1(f)} = \frac{P(f)|S(f)|^2}{|S(f)|^2} = P(f) \tag{2.14}
\]
2.2.2 Stabilization factor

When using a pre-whitening filter a problem arises when the source signal is very weak or absent. In this situation the CCF is divided by the auto power spectrum, which is nearly zero, making the transfer function potentially unstable. To prevent this a stabilization factor is introduced in the filter.

\[ H(f) = \frac{G_{x1x2}}{G_{x1x1} + A} \] (2.15)

Here \( A \) is the stabilization factor, which prevents division by zero (or a very small value). The value of \( A \) must be chosen so that it is small compared to a strong signal, and large compared to a very weak signal. Reeuwijk [1] suggested using a stabilization factor which is set at the level of the auto-correlation of the noise in the close microphone signal. This value will be used as a starting point for further optimization of the source tracking algorithm.

2.3 Peak detection

To determine the time delay, the position of the DSS peak in the transfer function \( h(t) \) (resembling the impulse response \( p(t) \)) must be determined. In general, this peak will be the first and highest peak in the transfer function \( h(t) \). Any reflections of walls and ceiling will show up as additional peaks after the DSS peak. Because these reflections will have traveled a longer path than the DSS, they will be attenuated.

In simple situations (no reflections present that could ‘hide’ the DSS peak), the highest peak detected in the transfer function will be assumed to be the DSS peak.

A problem arises when two or more reflection peaks coincide, and the total height of the reflection peak becomes higher than the DSS peak. In this scenario the DSS peak is no longer the highest peak, and the coinciding reflections are mis-detected as the DSS peak.

Another situation that poses a problem is when the source signal level is very weak. When the signal to noise ratio is not so good, it might prove to be difficult to extract the signal from the noise.

When faced with these less than ideal circumstances, we need a way to make the detection algorithm more robust. To do this we test the detected peaks against a number
of criteria to determine whether have found a valid DSS peak, or if we should reject the peak.

### 2.3.1 Noisy signals: Threshold value

When the source signal is weak relative to the noise, it becomes very difficult to accurately distinguish the DSS peak from the noise signal. Some peak-like features of the noise signal might be mistaken for actual peaks. And it might be possible that the noise contains a peak that is higher than the DSS peak. To avoid this, any peaks detected that do not exceed a certain threshold value are discarded. This includes the DSS peak when it is very weak. In essence, sensitivity is traded off for accuracy.

The higher the threshold value, the less chance there is of noise related peaks being detected. The downside is that the DSS signal must be quite strong to detect any peak at all. So the threshold value is an important parameter that must be tuned to find a good balance between sensitivity and accuracy.

The level of noise present in the signal is of course an important parameter in setting the correct noise threshold value. So, to find a suitable value for the threshold the standard deviation $\sigma$ of the entire signal is calculated, including any (significant) peaks. The peaks can safely be included in the calculation, because their influence on the value of the standard deviation is very small. Furthermore, a very accurate value of the standard deviation is not required. Finally, the threshold value is set at 2.7 times the standard deviation of the signal. At this value of the threshold almost no noise is mis-detected as a peak, but the value is still low enough to pass through any significant real peaks. This value is derived from visual inspection of the transfer functions and from simulations (see chapter 4), and should only be considered as a starting point for further optimization.

### 2.3.2 Coinciding reflections: Relative height

When two reflection peaks coincide, and form one large peak, the situation arises where the DSS peak is followed by a peak which is higher than the DSS peak. In this case the highest peak will result in incorrect determination of the position.

The assumption is made that the chance of more than two reflections coinciding is negligible. So we are only dealing with the situation where two reflections might
Chapter 2: Detecting the direct sound source peak

coincide. Furthermore, because the individual reflections are basically weaker than the DSS peak, the combined peak height of the coinciding reflections is not likely to be higher than twice the DSS peak.

The relative height of the highest peak (RHOHP) is defined as the ratio of the value of the highest peak $h_1$ and the value of the second highest peak $h_2$. The relative height method checks if the RHOHP exceeds a certain minimum value. As a starting point the minimum value is set at 1.7, as is suggested by Reeuwijk [1]. This means that the highest peak must be at least 1.7 times as high as the second highest peak to be accepted as a possible DSS peak. If the highest peak is less than 1.7 times as high as the second highest peak, it is rejected. Again, sensitivity is traded off for accuracy.

2.3.3 Stage size windowing

In general, the size of the stage area is known beforehand. The longest straight path a sound wave can travel onstage is determined by its size (and shape). For a rectangular shaped stage the distance between two opposing corners can be taken as the maximum length a sound wave can travel in a straight line.

Given this information we can reject any peaks that come after a certain maximum travel time. Any peaks detected corresponding to a position beyond the boundary of the stage area can be safely ignored.

2.4 Time window length

For a wave-field synthesis system to accurately amplify the sound of a moving source, it needs to have the position of the source updated several times per second. If the position can not be determined often enough, the sound will seem to ‘jump around’ when the source moves across stage. To make the successive updates unnoticeable to the audience, the position must be determined as frequently as possible.

If the goal is to detect the position, for example, ten times per second, the signals are divided into time windows of one tenth of a second (100 ms). The length of one time window is called the Time Window Length (TWL). All correlation and filtering operations are applied to one time window at a time. This will result in a single position being determined. So, if we are able to calculate the position within a TWL of 100 ms,
we can update the position ten times per second.

The shorter the TWL, the more often it is potentially possible to detect a position. The downside of a short TWL is that for each time window, the cross correlation function is based on fewer samples. The peaks in the CCF will therefore be less pronounced, and it might not be possible to accurately determine a clear DSS peak. To increase the probability of detection the TWL needs to be longer.

These two conflicting requirements must be carefully balanced to find an optimum between good sensitivity and fast detection.

Reeuwijk [1] suggested the use of a TWL of 80 ms, as this value theoretically optimizes the balance between sensitivity and speed of the system. But in practice 80 ms has proved to be too short, resulting in a system that fails to detect a lot of peaks. When the TWL is increased up to 250 ms the performance of the system improves significantly. So the TWL will initially be set at 250 ms.
Chapter 2: Detecting the direct sound source peak
3.1 Introduction

Given the delays at all the remote microphones, it is possible to determine a 2-d position on stage. The delays can easily be converted to distances of the source from the remote microphones using the speed of sound (which under normal conditions\(^1\) can be assumed to be 340 m/s). The position of the source can be calculated using these distances as described in the following sections.

3.2 Intersecting circles

When the distance from one of the remote microphones to the source is known, the source position must be somewhere on a circle with a radius equal to the calculated distance, and the remote microphone position as the center of the circle. When a second distance from another remote microphone position is known, the source must be located on the intersection of the two circles (see figure 3.1)\(^2\). In general, this will lead to two possible source positions, corresponding with the two intersection points of the two circles. To correctly determine the source position, we must now choose which

\(^1\)In air at 20°C

\(^2\)When the three sides of a triangle are known, it is possible to calculate the angles using the cosine rule (see appendix B). From here on it is quite straightforward to calculate the position, since the remote microphone positions (which form two of the corners of the triangle) are known.
one of the two solutions is the correct one, and which one we should discard. For a remote microphone configuration consisting of a linear array at the front of the stage, this is fortunately quite simple. The position of the two intersections lie symmetrically on either side of the array, and since the array also forms the front boundary of the stage area, one of the intersections will always be in front of the stage area, and therefore falls outside the stage area. So, in this case we can simply discard this solution.

![Intersecting circles](image)

**Figure 3.1** Intersecting circles
3.3 Stage area

3.3.1 Rejection of off-stage positions

After we have rejected the solutions which lie in front of the array, there is only one possible solution left. If this position is not within the pre-defined stage area, which may be caused by incorrect detection of the DSS peak, it must also be rejected.

3.4 Average position

As was explained in section 3.2 a combination of two remote microphones is required to obtain one source position. There are \( \frac{1}{2}(n^2 - n) \) possible unique combinations between two remote microphones, where \( n \) is the number of remote microphones that have detected a valid DSS peak. Each of these combinations will result in a position, if the position calculated falls within the stage area. In the ideal situation where all remote microphone positions are exactly known, as is the speed of sound, and the DSS peak is detected accurately, the detected positions will all coincide. This is usually not the case. Therefore the detected positions will form a ‘cloud’ of points around the true source position. To derive the final output of the source-tracking system, the detected positions are averaged by taking the average value of the x and y coordinates of the detected positions.

3.5 Systematic error

When detecting the distances, the remote microphones and the close microphone are all assumed to lie in a horizontal plane. This reduces the problem from three dimensions to two dimensions. In reality, this is not true. The remote microphones are placed on the ground at the front of the stage, while the close microphone is positioned near the mouth of the speaker. Thus the detected distance is the three dimensional distance from the remote microphone to the mouth, while the calculations performed here assume a two dimensional plane. As can be seen from figure 3.2 this effect causes the detected distances to be larger than the actual distance. This effect is most obvious when the speaker is positioned relatively close to the remote microphones, like when the speaker
Chapter 3: Determining onstage position

Figure 3.2  Systematic error due to neglected speaker height.

walks up to the front of the stage. When the speaker walks toward the back of the stage, this effect is less pronounced.

It is possible to correct for this deviation by using the following formula:

\[ d' = \sqrt{d^2 + h^2} \]  (3.1)

where \( d \) is the detected distance, \( d' \) is the actual distance and \( h \) is the speaker height. However, the height of the speaker is not known. It might be possible to measure the speaker height, and manually enter it into the system, or alternatively assume an average height for the speaker. But this correction then no longer holds when the height is not constant. This situation could occur for example when the actor or singer sits down.
Chapter 4

Simulation

4.1 Introduction

In the early stages of the development of this source-tracking system, simulations were used to evaluate the performance of this system [1]. It also serves to find some initial values for the parameters (like the TWL, stabilization factor, etc) used in the system. These values are only a starting point for the parameters used with real world data. When this data is available, further fine tuning of the parameters will be possible.

The use of simulated data was necessary because real world data was not yet available. Some sound samples were recorded, and processed to simulate real world data. These processed samples formed the test input for the source-tracking system under development.

The unprocessed sound sample formed the input for the close microphone. The same data was then processed to simulate the input at the remote microphones.

4.2 Remote microphone signal simulation

4.2.1 Geometrical spreading

Sound radiated by a monopole source is attenuated due to geometrical spreading. This spreading is a function of the distance between source and receiver, and accounts for $1/r^2$ of attenuation, or $20 \log r$ in dB’s. To simulate geometrical spreading the test
sample is attenuated by a number of dB’s according to the distance between the (virtual) close microphone and remote microphone.

### 4.2.2 Travel time delay

Compared to the close microphone signal, the signal at the remote microphone is delayed by a certain amount, which is also a function of the distance between the close microphone and remote microphone and the speed of sound. The amount of delay is the parameter which the source-tracking system will try to determine. To simulate this, the sample is transformed into the Fourier domain, and multiplied by $e^{-j\omega d/c}$ according to the distance $d$ between the close microphone and the remote microphone, and the speed of sound $c$. Afterwards, it is transformed back into the time domain, resulting in a signal that is delayed by $d/c$.

### 4.2.3 Noise

In real life applications noise is almost always present. So the design of this system must be robust enough to handle some amount of background noise to operate properly.

To simulate real world ambient noise white noise is added to the remote microphone signal. This is one of the most important effects that influence the performance of the source tracking system. Without noise the system is capable of almost perfect source tracking, but when noise is present the performance is negatively affected, as it becomes more difficult to extract the speech signal from the noise.

No noise is added to the close microphone signal. Because it is located so close to the source any noise picked up will be negligible.

### 4.3 Evaluation

By varying the amount of attenuation and delay, a virtual stage geometry can be simulated, and the performance of the source-tracking system can be tested with various amounts of background noise, and with various signal types (noise, speech, etc.). Also, the effect of the pre-whitening filter can be evaluated at various settings of the stabilization factor.
To quantify the performance of the system a relevant parameter must be found. For this purpose the number of tracking ‘dropouts’ occurring during a source tracking session can be counted. Few dropouts corresponding to good source tracking performance. A more refined method for measuring the performance is to count the number of microphone pair combinations that yield a position value. When the signal is strong, a large number of remote microphones will show a large and clear peak in the cross correlation function, and therefore a large number of remote microphone pair combinations will yield a valid position value. When the signal is weak, and only picked up by a small number of remote microphones (probably located relatively close to the source), only a small number of remote microphone pairs will be valid. This is a result of weak signals being discarded by the system. When the signal is very weak, it will not be picked up by any remote microphone. This will result in a dropout. When the signal is strong enough to be picked up by some remote microphone, but not all microphones, the number of valid microphone pairs can be seen as the amount of ‘confidence’ in detecting the source position, with \( \frac{1}{2}(n^2 - n) \) being the maximum number of possible combinations using \( n \) remote microphones.

\[
\text{Figure 4.1} \quad \text{Top: Source signal (speech). Bottom: Number of valid remote microphone pairs detected. Notice that between words (where the source signal is very weak) the detection drops to zero.}
\]
4.4 Conclusions

The results from the simulations show that in theory this approach to source tracking speech signals works, as was already concluded by Reeuwijk [1]. The performance of the system is however very much a function of the signal-to-noise ratio at the remote microphones.

When the signal is weak compared to the ambient noise, the choice of values for the various parameters is important to make the system able to extract a correct position. Table 4.1 lists the values for the parameters used as an initial starting point for further optimization. These values were found in part by visual inspection of the cross correlation functions, and also through trial and error.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Window Length</td>
<td>80-250 ms</td>
</tr>
<tr>
<td>Noise Threshold</td>
<td>2.7</td>
</tr>
<tr>
<td>RHOHP</td>
<td>1.7</td>
</tr>
<tr>
<td>Stabilization factor</td>
<td>0.5</td>
</tr>
</tbody>
</table>

*Table 4.1 Initial parameter values*
Chapter 5

Real world implementation

5.1 First prototype

In figure 5.1 the configuration of the non-real time prototype is shown schematically. It consists of eight microphones, seven of which are used as remote microphones. These microphones are connected to an A/D converter, which also supplies phantom power to the microphones. The digital signal is then recorded on a multitrack tape. Finally, a digital sound card is used to transfer the recordings to a computer.

5.1.1 Microphones

Seven B & K condenser microphones with a cardioid sensitivity pattern were mounted on floor stands, about 10 cm from the ground. They were placed along the front edge of the stage area (see figure 5.2). The output of the microphones is converted by a Tascam A/D converter into a digital TDIF signal. This signal then forms the input for the digital tape recorder. The A/D converter also supplies phantom power to the condenser microphones.

5.1.2 Digital tape recorder

In the initial setup it was not possible to directly read the audio data into a computer to do real-time source tracking. The signals were first recorded using a Tascam 8 track digital tape recorder (see figure 5.3). The fact that this recorder can record up to
eight tracks limits the number of channels that the system can use. The first channel is used to record the direct sound from the close microphone. The remaining seven channels are available for the remote microphones. This makes it possible to calculate up to seven cross-correlation functions simultaneously during later processing of the recorded data.

### 5.1.3 Sound card

To transfer the recordings from the tape drive to the network, a PC equipped with special sound hardware is used. This consists of a Tascam PCI-822 sound card. This sound card has a TDIF digital audio input. Using this hardware it is possible to transfer up to eight channels of digital audio simultaneously into the computer from the digital.
tape recorder, and therefore calculate up to seven cross correlation functions.

5.1.4 Software

After the sound data was transferred into a computer it is saved as standard wav files, at 44.1 kHz sampling rate and 16 bit resolution. To do the actual processing (cross-correlations, peak finding, etc.) a program was developed using Matlab.

During the initial development of the source tracking software, Matlab proved to be a valuable tool. It is easy to visualize the intermediate results from every stage of the source tracking process. Also, the results of changes in the program can very quickly be evaluated, since no recompilation is necessary.
Chapter 5: Real world implementation

Figure 5.3  The Tascam DA-88 digital multitrack tape recorder.

The downside is that because the program code is not compiled but interpreted it is inherently slow. Another problem with Matlab is that Matlab itself is a single-threaded application. This makes it unable to use the extra processing capabilities of SMP\(^1\) computers. The source tracking algorithm described in this thesis contains a lot of potential parallelism, which could be exploited on SMP machines. Finally, it was not possible to read the audio data from the sound card directly into Matlab in real-time. Because of these reasons the final implementation of the source tracking system does not make use of Matlab.

5.2 Real time implementation

5.2.1 Hardware

The availability of new hardware opened up new possibilities for the second (real time) implementation of the source tracking prototype, the configuration of which is shown schematically in figure 5.4. This new hardware is far more suitable for implementing a real time (audio) signal processing system, like source tracking. It consists of a Reference PC\(^8\) (see figure 5.5) built according to specifications from RME-Audio

\(^1\)Symmetric Multi Processing: A type of computer containing more than one CPU, making it potentially faster for programs capable of utilizing the extra CPU’s.
(see table 5.1).

**Figure 5.4** Configuration of real time prototype.

<table>
<thead>
<tr>
<th>Case</th>
<th>Full tower case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motherboard</td>
<td>MSI 694D Pro (MS-6321) based on VIA 694XDP chipset</td>
</tr>
<tr>
<td>CPU</td>
<td>2x Intel Pentium III (Coppermine) 1 GHz with 256 kB L2 cache and 133 MHz FSB</td>
</tr>
<tr>
<td>Harddisk</td>
<td>2x IBM DTLA-307030 Deskstar 75GXP ATA100, 30 GB, 7200 rpm, 2 MB cache</td>
</tr>
<tr>
<td>Memory</td>
<td>256 MB PC133 2-2-2 SDRAM</td>
</tr>
<tr>
<td>CD-ROM</td>
<td>Asus CDS-500</td>
</tr>
<tr>
<td>CD-Writer</td>
<td>Teac CD-W512E</td>
</tr>
<tr>
<td>Graphics</td>
<td>Matrox G400/450 Dual Head 32 MB</td>
</tr>
</tbody>
</table>

**Table 5.1** Reference PC: Dual CPU Audio Workstation specifications
The most interesting part of these specifications is the fact that this is a dual processor SMP machine. Having two CPU’s available for processing the source tracking algorithm makes it possible to run the source tracking system in real time (comfortably), even using more channels than was previously possible. The interface between the CPU and the chipset (Front Side Bus) seems to be a bottleneck in this system, however. It limits the amount of data that can be transferred from memory to the CPU’s and back. This might prove a problem when trying to calculate large FFT’s (larger than the size of the CPU cache). In the case of the source tracking system, it does not seem to be a problem, since the FFT sizes are relatively small. An FFT consisting of 8192-24576 elements is a typical size used in the source tracking system.

The workstation forms the platform for the RME Hammerfall digital audio interface[7]. This sound card is capable to simultaneously record and playback up to 26 channels.
24 of the 26 input channels are formed by 3 ADAT interfaces, each consisting of 8 channels, using ‘lightpipe’ optical fibers. Each channel can capture (or play back) audio data at 44.1-48 kHz sampling rate, and 24 bit resolution. The remaining two input channels are S/PDIF interfaces.

5.2.2 Microphones

The microphones used are the same Brüel & Kjær 4011 condenser microphones with a cardioid sensitivity pattern as in the first prototype, also mounted on floor stands. Up to eight microphones can be used per TDIF/ADAT channel. So, the Hammerfall sound card could theoretically capture 24 microphone signals. But because of the limited number of microphones available only eight microphones were used in this implementation of the source tracking system.

5.2.3 Interfacing digital audio equipment

<table>
<thead>
<tr>
<th>Microphones</th>
<th>Bruel &amp; Kjaer cardioid condenser microphones type 4011</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/D converter</td>
<td>Tascam MA-AD8 (20 bit, 44.1/48 kHz),</td>
</tr>
<tr>
<td></td>
<td>Tascam TM-D4000 (24 bit, 44.1/48 kHz)</td>
</tr>
<tr>
<td>TDIF-ADAT converter</td>
<td>Tascam IF-TAD</td>
</tr>
<tr>
<td>Multitrack recorder</td>
<td>Tascam DA-88 (16 bit, 44.1/48 kHz)</td>
</tr>
<tr>
<td>PC sound interface</td>
<td>RME Hammerfall DIGI9652</td>
</tr>
<tr>
<td></td>
<td>Tascam PCI-822 (used in non-real time system)</td>
</tr>
</tbody>
</table>

Table 5.2 Audio equipment

Since the inputs on the Hammerfall sound card are all digital, external A/D converters must be used to capture the analog microphone signals. For this purpose the same Tascam hardware was used as in the first prototype. One Tascam MA-AD8 A/D converter, capable of supplying phantom power to the condenser microphones, was available. Additionally, a Tascam TM-D4000 (see figure 5.6) digital mixing console can be used as an A/D converter, adding eight additional channels. Interfacing the Tascam equipment with the Hammerfall sound card proved to be a challenge, however, since
the TDIF format used by Tascam is not compatible with the ADAT format used on the Hammerfall sound card. To overcome this problem Tascam IF-TAD format converters are used to convert between the TDIF and ADAT formats. Unfortunately, the IF-TAD format converters are unable to pass through the synchronization signals from one format to the other. Therefore, external word clock (coax) cables must be used to transfer the synchronization signals to all the digital equipment in the setup\(^2\).

![The TM-D4000 digital mixer, used as an A/D converter.](image)

**Figure 5.6** The TM-D4000 digital mixer, used as an A/D converter.

### 5.2.4 Software

The software part of the source tracking system was written for the GNU/Linux operating system. This platform was chosen because the source code of the operating system is freely available for download from the internet\(^3\), making it easy to make modifi-
cations to the kernel. More specifically, the Linux kernel was modified\(^4\) (patched) to have improved real time performance.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Red Hat Linux 7.2([12])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>Linux 2.4.19-pre7([11]) with Robert M. Love’s preempt patch and Andrew Morton’s low latency patch</td>
</tr>
<tr>
<td>Sound driver</td>
<td>Alsa 0.9rc1([9])</td>
</tr>
<tr>
<td>FFT library</td>
<td>Fastest Fourier Transform in the West 2.1.3 (FFTW)([10])</td>
</tr>
<tr>
<td>User interface library</td>
<td>Trolltech Qt 3.0.3([18])</td>
</tr>
<tr>
<td>Compiler</td>
<td>GCC 2.96</td>
</tr>
</tbody>
</table>

\[4\] Clark Williams\([15]\) of Red Hat, Inc. describes in his article the effects of applying the preempt and low latency patches to the Linux kernel.

The Hammerfall sound card is controlled by the Alsa drivers\([9]\). These drivers, together with the accompanying libraries, provide a convenient abstraction layer between the hardware and source tracking software.

The source tracking software itself is written in C++. It uses the Qt libraries from Trolltech for the graphical user interface. The software draws a graphical representation of the stage area on screen (see figure 5.7). A small square represents the current position of the source, as calculated by the source tracking algorithms. To the left of the stage area an indicator shows the percentage of detected positions that were averaged (and thus not rejected) for calculating the final source position.

The second screenshot in figure 5.8 shows where the various parameters can be set in the program. Settings like the positions of the remote microphones can be saved to disk to avoid having to fill in all positions over and over again.

The source tracking program makes extensive use of POSIX threads for splitting up the computation in a number of ‘threads’. Each thread can be executed independently of the other threads, and on SMP computers threads can be executed simultaneously. By splitting the algorithm in a number of threads the workload can be spread over the two CPU’s, significantly speeding up the process.
Figure 5.7 Screen shot of the ‘stage area’. The red square in the middle represents the current source position on the stage area (the blue field). The red bar on the left is a meter that displays the number of detected positions averaged to derive the final position, with the full scale being 21 positions when using 7 remote microphones (see section 3.4).
Figure 5.8  Screenshot of the settings page. Here the various parameters can be entered, like the size of the stage, the time window length, the sampling rate, etc.
Chapter 6

Results

6.1 First prototype results

By recording some signals on a digital multitrack recorder, and analyzing the results on the computer it is possible to test the algorithm with various parameter settings.

6.1.1 Number of remote microphones

To find out how the performance of the source tracking algorithm depends on the number of remote microphones in use, the same recording is processed several times, while varying the number of remote microphones that are actually processed. The configuration of the remote microphones is shown schematically in figure 6.1.

![Remote microphone configuration](image)

*Figure 6.1  Remote microphone configuration.*
As can be seen from figures 6.2 through 6.4 the number of remote microphones used has a profound impact on the performance of the source tracking system. The remote microphones were along the front edge of the stage area as usual, spread out along the entire width of the stage. In the cases where not all remote microphones were used, only the first 3 or 5 microphones from the left were used. Figures 6.3 and 6.4 show that only when the source was sufficiently close to the active microphones the system was able to determine a position. This might indicate that the system is only capable of generating a cross correlation function with a clear DSS peak when the source is within the (limited) range of a remote microphone. The range is determined by the signal to noise ratio of the speech signal recorded by the remote microphone. Also, a certain degree of redundancy in the number of remote microphones must be present, as fig. 6.4 shows that even when the source is relatively close to the active microphones the performance is very bad when only a small number of microphones is active.
Figure 6.3  The same recording as in fig. 6.2, but using only microphones 1-5. Notice that the right part of the source path was not detected.

6.1.2 Stabilization factor

By processing the same recorded signals with different values for the stabilization factor, which was introduced and discussed in section 2.2.2, the effect of this parameter can be tested. Figures 6.5 through 6.8 show the same signal processed with $A = 0.5, 0.25, 0.1, 0.05$, all using 7 remote microphones.

This experiment shows the effect of choosing a stabilization factor that is too small. As was pointed out in subsection 2.2.2 the use of a pre-whitening filter introduces the risk of the filter becoming unstable. This will most likely occur when the speaker is not producing any sound, like between words. The purpose of a pre-whitening filter is to ‘manipulate’ the spectrum of the signal to produce a white spectrum, as this in effect will make the peaks sharp (and therefore easier to detect). It does this by amplifying some frequencies, and attenuating others.

When the microphone signal is weak or absent, only the noise is left, which is amplified by the pre-whitening filter. This introduces features in the cross correlation function that are misdetected as valid peaks. The position of these peaks, of course, have no
relation to the real position. The end result is that the system is outputting position data that is nowhere near the true source position. In fact, these peaks have managed to pass the tests that are designed to filter out these false peaks.

To prevent pre-whitening filter instability the stabilization factor was introduced. This prevents the filter from becoming unstable when the signal is weak or absent, but it must have minimal effect when the signal is strong. So, one must be careful not to choose the stabilization too large, as this will ‘color’ the spectrum of the cross correlation function, and result in peaks that are no longer sharp. These wide peaks could present problems for the peak detection algorithm. Choosing the stabilization factor too small, however, allows the filter to become unstable, as can be seen in figures 6.7 and 6.8. In this situation it seems that the optimal value of the stabilization factor lies somewhere between 0.1 and 0.25. It is safe, though, to use a stabilization factor that is somewhat larger than strictly necessary, like 0.5 in figure 6.5.

Figure 6.4  Again as in fig. 6.2, but now with microphones 1-3. Again, the right-hand part of the path was not detected.
Figure 6.5 Stabilization factor $A = 0.5$.

Figure 6.6 Stabilization factor $A = 0.25$. 

Chapter 6: Results
Chapter 6: Results

Figure 6.7 Stabilization factor $A = 0.1$.

Figure 6.8 Stabilization factor $A = 0.05$. 
6.2 Real time results

6.2.1 Time Window Length

As was concluded earlier in chapter 2 a TWL of only 80 ms contains too little data to calculate a cross correlation function which is sufficiently clear to extract a DSS peak. Increasing the TWL to 250 ms significantly improves the detection performance of the source tracking system. The trade-off, however, is that now only 4 positions per second can be calculated instead of 12.5 (1000/80). Also, the amount of computation required to calculate the FFT’s is much larger for 250 ms time windows because FFT scales as $O(N \log_2 N)$. However, neither the 80 ms or the 250 ms time windows result in desirable performance, as 80 ms does not result in enough usable positions, and 250 ms limits the position update rate to 4 per second. In the non-real time implementation the default setting for the TWL was 250 ms. This resulted in acceptable source tracking performance. Improving the low update rate was not a first priority at the time.

The first step to improve the performance is to restrict the time window lengths to have $2^n$ samples (where $n$ is an integer). This is because the FFT library used can be about 1000 times faster when the FFT length is an integer power of 2. The performance is very low for FFT sizes which contain large prime factors. So these sizes should be avoided at all costs. So instead of choosing 250 ms for the time window length (which corresponds to 11025 samples at 44.1 kHz sampling rate), it is much more efficient to choose 186 ms or 372 ms (corresponding to 8192 and 16384 samples at 44.1 kHz sampling rate). The restriction can be loosened somewhat to include sizes consisting of simple factors, like $2^a3^b$, without sacrificing too much speed. So choosing a size of 24576 samples is still quite fast, since $24576 = 3^12^{13}$, and it will compute faster than $32768 = 2^{15}$.

A TWL of 24576 samples is equivalent with 557 ms, which is sufficiently large to contain enough data to produce DSS peaks which are easy to detect. Also, this size is still comfortably within the computational capabilities of the hardware. The only problem seems to be that with a time window of 557 ms less than two time windows per second can be calculated. Fortunately, a solution to this problem will be presented

---

1The FFTW library is most efficient with integer powers of 2 FFT lengths, but is still quite fast with lengths of $2^a3^b5^c7^d11^e13^f$ where $e + f$ is 0 or 1. For other FFT sizes computation is slower, and performance could even be reduced to $O(n^2)$ for prime sizes.
6.2.2 Time window overlap

To unite the requirements of having large time window lengths and a high rate of detection the time windows are made overlapping. This is possible when enough computation power is available to calculate one time window in less time than the length of the time window. If, for example, the computation of one time window of length 557 ms (24576 samples at 44.1 kHz) takes 100 ms to complete, then instead of waiting for the next 557 ms of data to be available only 100 ms of new data is used together with 457 ms of old data. This will result in 10 positions per second calculated, while still using a large time window (see fig. 6.9).

The effects of using extremely long time windows have yet to be tested however, because during the 557 ms the source could move some distance across stage. The position reported by the system could be any point along the path traveled by the source during the time window. Or it could possibly be some sort of average position during the time window. Also, the detection capability could possibly be negatively affected by source movement during a time window due to Doppler frequency shift\(^2\).

\(^2\)Assuming that the source speed doesn’t exceed 3.4 m/s (a fast walk) the frequency shift should be less than 1 %. The effect on the cross correlation function is not known however.
6.2.3 Array configuration

To finally test the real time source tracking system a microphone array was laid out in an empty room. The room dimensions are approximately 6 m x 4 m. The remote microphones were spread out along the front edge of the ‘stage area’, approximately 75 cm apart. Seven remote microphones were used, along with one close microphone, totaling eight microphones. This number is the maximum that can be used using just one A/D converter. Also, this was the number of microphones that were available at the time. In this case the TM-D4000 mixer was used as the A/D converter and microphone preamp.

Using this configuration, a person walking around in the room can be tracked on the computer located in the adjacent room. As long as the source is speaking into the close microphone, the little square on the computer screen will track the movements of the source (see figure 5.7).
Chapter 7
Extrapolation

7.1 Introduction

In case the source-tracking system is unable to detect a position during some time-window, Reeuwijk [1] suggested that the positions detected during the previous time-windows could be used to extrapolate the current position. The most simple case would be 0’th order extrapolation, or 0’th order hold. In this case the previous position is assumed as the current position when we are unable to detect a position. This position is held until the system has detected a new position. Actually, this can hardly be called extrapolation at all.

The problem with 0’th order hold is that when detection is lost for a short period of time while the source is moving across stage, the last known position is held until a new position can be detected. However, during this silent period the source has moved some distance, and the detected position will seem to have jumped from the last known position to the new position.

A more sophisticated approach is first order or linear extrapolation. In this case the speed of the source is taken into account when extrapolating to a new position. To calculate the new position, the previous two positions are used to determine the speed and direction of the source. The current position is then estimated based on the previous position and the previous speed and direction. In theory this will bring the last known (extrapolated) position closer to the true position during periods of detection loss.

Reeuwijk [1] even suggested using higher order extrapolation to compensate for the
fact that the source might not be walking in a straight line at a constant speed. A second order extrapolation algorithm could be better at estimating the current position when the source is following a curved path because not only the speed and direction, but also the acceleration and changes in direction are considered.

### 7.2 Results: Detection runaway

In order for linear extrapolation to work, the assumption must be made that only a few time-windows are not detected. When almost all timewindows are correctly detected, the few undetected time-windows (dropouts) can be estimated by using linear extrapolation. This approach fails to work when the detection rate falls significantly below 100%.

In a typical situation the detection rate is closer to 50% than it is to 100%. This means that only once every couple of time-windows a position can be detected. Figure 7.1 shows the result of linear extrapolation under these conditions. The estimated positions ‘run away’, only to jump back whenever the next time-window is correctly detected. These runaways cause the linear extrapolation technique to yield far worse results than no extrapolation. Figure 7.2 shows the same data without extrapolation (0’th order). It is clear to see that in this case the source path is followed more accurately. The current position however still jumps from the previously detected position to the next, but it no longer leaves the path of the source.

### 7.3 Conclusion

When the performance of the source tracking system is close to 100% (very few dropouts) first or second order extrapolation might be suitable to fill in the missing positions. But when the performance drops significantly below 100% the higher order extrapolation techniques exhibit ‘run away’ behavior, making them unsuitable for estimating the source position. Zeroth order extrapolation, however, is safe to use regardless of the performance of the system.

In real world implementations of the source tracking system the performance is in general significantly less than 100%. This makes zeroth order extrapolation a far safer choice over any higher order extrapolations.
Chapter 7: Extrapolation

Figure 7.1  First order extrapolation

Figure 7.2  Zeroth order extrapolation
8.1 Introduction

The source tracking method as discussed so far performs adequately when the source signal is reasonably strong. When the source signal is weak, or when there is a lot of background noise present, the performance is degraded. The performance of the system depends a lot on the signal strength. This is a consequence of the many ‘tests’ the system performs that decide whether to use a certain signal/channel or to drop the data. These tests reject bad or weak data to ensure that only strong signals are used to correctly determine the source position. When the source signal is too weak, all data might be rejected, leaving the system unable to determine a source position.

During this process weak signals that do not pass these test are simply discarded. These signals, however, might still contain some information, buried deep in the noise. The inversion method, which is commonly used in the field of seismic exploration, is especially suited to extract weak signals from noisy data.

8.2 Theory

The principle behind the inversion method is not to try to determine the position directly from the cross correlation data, but to determine for every possible source position the likelihood that it is the true source position.

The stage area is divided into a grid. For every point on this grid the source position
likelihood is calculated. The distance between grid points determines how accurately the source position can ultimately be determined. Shorter distances between grid points result in a higher resolution ‘picture’ of the stage area. However, since a lot of computation is required for every grid point, using a very fine grid mesh will result in very long computation times, and the final choice of grid resolution must be balanced between high resolution and limited computation time.

To determine the likelihood of a source position for a grid point its distance to all the remote microphone positions are calculated. Using the speed of sound these distances are converted into delays. The next step is to shift the cross correlation functions (to the left) by their respective delays, and by doing this moving any peaks present in the cross correlation function. When the grid position under consideration is indeed the true source position the large DSS peak present in the cross correlation functions will all line up at $t = 0$, like in figure 8.2. The next step is to add up for all $t$ the shifted cross correlation functions. This will result in a very large peak at $t = 0$ when all DSS peaks indeed line up (figure 8.3), but will show a much smaller value at $t = 0$ when the DSS peaks do not line up which is the case when the grid position is not the source position (see figure 8.4).

One thing that must also be taken into consideration is that because the stage area is effectively sampled at some spatial resolution (for example every 10 cm), it is very likely that the source position is not exactly equal to a grid position, but might fall somewhere between 2 grid positions. This means that the DSS peak does not line up exactly at $t = 0$. The shifted DSS peaks are more likely to cluster at a small

\[ \text{Figure 8.1 Cross correlation functions before lining up.} \]
interval around $t = 0$. So, to calculate the likelihood of a source position at a grid position the energy around a small time interval around $t = 0$ is taken (the time interval might be taken to correspond to the distance between grid points). By doing this the algorithm can account for the DSS peaks not lining up perfectly, without sacrificing any precision.

When the stage area is systematically scanned using the inversion algorithm it will result in pictures of the stage like in figures 8.5 through 8.8. Figure 8.5 shows the
inversion method applied to a situation with clear signals. In this case it is straightforward to find the source position, as it is at (or very near) the grid point with the highest value (colored red). Figure 8.6 shows an example of a situation which the CCF is not so clear. It has several areas with high probability values (the red areas). But in this case it is not so easy to determine which of the red areas is the true source position. The third example shown in figure 8.7 has one remote microphone channel with a very clear CCF. The other channels however do not have a sufficiently clear CCF to accurately determine the intersection of the circles. The last example in figure 8.8 shows so many circle like features that it is quite impossible to easily determine a position.

8.3 Results

In situations where the normal source tracking method is unable to produce any results, the inversion method is also unable to produce a clear result. In this case the results are too ambiguous to accurately determine a position. When the inversion method yields clear pictures, it is usually also quite easy for the conventional algorithm to detect a position. Also, the inversion method is much more CPU-intensive, and therefore not really suited for real-time source tracking. The inversion method, however, has some potential when applied in a more optimized form where not all grid points are scanned.

Figure 8.4  Summing up peaks that do not line up, resulting in a low value for the sum.
Figure 8.5  Inversion method used an a very clear signal. Notice the intersecting circles can actually be seen in this picture.

Figure 8.6  The CCF does not seem to have a clear DSS peak in this picture.
Figure 8.7  Although the circle like features are very clear in this picture, it is not clear where the source was actually located

Figure 8.8  Another example that is not suitable for detecting the source position
but only the area around a known previous source position. It could also be useful for visually verifying the results from the normal source tracking system.
Chapter 9

Conclusions and suggestions for further research

9.1 Research goals

In chapter 1 it is stated that the goal of this research is to implement an on stage source tracking system, based on the research by Reeuwijk[1]. Starting with Reeuwijk’s M.Sc. paper, which contains the theory and some simulation results, an implementation of the source tracking system was created. During the course of the research period this system has evolved from the simulation stages to a working demonstration prototype, capable of tracking on stage sources in real time. Furthermore, the source code of this real time source tracking system can be used to integrate a component in a larger system, where the output from the source tracking part is used to control the virtual source position created by a wave field synthesis system.

9.2 Comparison of conclusions

Because the work done by Reeuwijk forms the base for the research in this thesis it should be interesting to re-evaluate some of the conclusions drawn by Reeuwijk in his M.Sc. thesis[1] using the knowledge gained from this project.
9.2.1 Time window length

Reeuwijk stated that to track a source moving across stage with a speed of 2 m/s an update rate of at least 10 Hz must be attained. This follows from the assumption that a person can just notice a source moving from left to right when the azimuthal angle is more than 2 degrees. This translates to a movement of 0.20 m when the spectator is sitting 6 meters from the stage. From this requirement the optimum time window length is specified as 80 ms, as this value should result in an optimal update rate. Implicitly the assumption is made that the detection rate is close to 100 %. We know now that this is generally not the case (especially when using 80 ms time windows). This is of course without overlapping time windows. By using overlapping time windows, the detection can be improved significantly by increasing the time window length, while still maintaining a fast update rate. The only requirement is the availability of a fast computer.

9.2.2 Number of microphones

In practice, the minimum number of remote microphones required is much larger than the three required to derive a unique position, assuming ideal circumstances. Using only three microphones the performance is very poor, as the source can only be tracked when in close range of the remote microphones. This is because the signal to noise ratio is very low at long distances. But even when the source is in close range the tracking is not very good when using a minimal configuration. Some degree of redundancy must be present to achieve acceptable performance. Using at least seven remote microphones spread along the front edge of the stage will yield far better results than the minimal configuration. On larger stages it might even be necessary to use more than seven remote microphones to ensure that the source is always within range of the microphones.

9.2.3 Extrapolation

Extrapolation of non-detected time windows may seem like a good idea in theory, but in practice the result is far worse than without extrapolation. This is because extrapolation can only fill in occasional dropouts in the detection. When more than a few time windows are not detected, extrapolation will result in detection runaway.
This leaves the problem of detection during silences unanswered. The problem is particularly bad when the speaker moves a great distance during a long period of silence. Some experiments should be conducted to determine the effects on the perception when the initial position of the amplified sound seems to jump from the old position to the current position when the speaker starts talking again.

9.3 Conclusions

- Time windows of 250 ms or longer are required to properly detect the source position.

- Time window overlapping should be used to increase the update rate while being able to use long time windows.

- To avoid severe performance degradation of the FFT library do not use FFT sizes consisting of large primes. Best performance is achieved when the FFT size can be decomposed in simple factors like $2^a3^b$.

- Considering the previous three conclusions a time window length of 24576 samples (at 44.1 kHz) should be a good starting value for further experimentation.

- At least seven remote microphones, positioned at 0.6-1.0 m intervals, are required for reliable tracking on a small stage. More microphones are probably required for larger stage areas.

- The stabilization factor $A$ should not be chosen too small. A value of $A = 0.25 - 0.5$ is a good starting point.

- Extrapolation, other than 0’th order hold, should not be used due to the risk of detection runaway.

- Inversion based methods could be useful for visual verification of the accuracy of the system, but are far too slow for real time implementations.
9.4 Suggestions for further research

- Find optimal values for parameters like the stabilization factor, noise threshold, etc.
- Apply more advanced filtering/signal processing to the input signals.
- Try to extract ‘hidden’ information from otherwise discarded data.
- Determine if using more microphones (along the side and rear edges of the stage) will improve tracking performance.
- Determine the optimal positioning of the remote microphones given a certain stage geometry and a limited number of microphones.
- Find a way to handle unknown positions after long silences. The position of the remote microphone with the strongest signal could be used to quickly determine a rough estimate of the source position after a long period of silence (a close and remote miking hybrid system).
- Evaluate the accuracy of the system.
- Evaluate the effect of using a different sampling rate. Sampling at 48 kHz instead of 44.1 kHz will result in more data for a given time interval, possible resulting in peaks in the cross correlation function that are more clearly defined.
- Improve the system performance when high levels of ambient noise are present.
- Explore the possibilities for tracking multiple sources on stage.
- Integrate the source tracking system with a wave field synthesis system.
Appendix A

Noise signals

The close and remote microphone signals contain noise $n_1(t)$ and $n_2(t)$.

$$x_1(t) = p_1(t) \otimes s(t) + n_1(t) \quad \text{(A.1)}$$
$$x_2(t) = p_2(t) \otimes s(t) + n_2(t) \quad \text{(A.2)}$$

Transformed into the frequency domain this becomes:

$$X_1(f) = P_1(f) S(f) + N_1(f) \quad \text{(A.3)}$$
$$X_2(f) = P_2(f) S(f) + N_2(f) \quad \text{(A.4)}$$

The impulse response at the close microphone position can be approximated by a Dirac delta pulse $\delta$.

$$p_1 \approx \delta \quad \text{(A.5)}$$

The Fourier transformed delta pulse has a white spectrum.

$$|P_1|^2 = 1 \quad \text{(A.6)}$$

The noise signals present in the close and remote microphone signals can be assumed to be uncorrelated.

$$G_{n_1n_2} = 0 \quad \text{(A.7)}$$

Because of this the noise signals disappear from the cross correlation between $x_1$ and $x_2$.

$$G_{x_1x_2} = X_1^* X_2 = (P_1 S + N_1)^* (P_2 S + N_2) = P_1^* P_2 |S|^2 + (P_1 S)^* N_2 + N_1^* (P_2 S) + N_1^* N_2 \quad \text{(A.8)}$$
Appendix A: Noise signals

\[ G_{x_1x_2} = P_1^* P_2 |S|^2 \]  \hfill (A.9)

However, in the auto correlation of \( x_1 \) it is still present.

\[ G_{x_1x_1} = X_1^* X_1 = (P_1 S + N_1)^* (P_1 S + N_1) = P_1^* P_1 |S|^2 + (P_1 S)^* N_1 + N_1^* (P_1 S) + N_1^* N_1 \]  \hfill (A.10)

\[ G_{x_1x_1} = |P_1|^2 |S|^2 + |N_1|^2 \]  \hfill (A.11)

Therefore the transfer function still contains the noise signal.

\[ \frac{G_{x_1x_2}}{G_{x_1x_1}} = \frac{P_1^* P_2 |S|^2}{P_1^* P_1 |S|^2 + |N_1|^2} \]  \hfill (A.12)

But since the close microphone is located so close to the source, the noise \( N_1 \) can be disregarded with respect to \( P_1 \), resulting in the following transfer function:

\[ \frac{G_{x_1x_2}}{G_{x_1x_1}} = \frac{P_2}{P_1} \]  \hfill (A.13)
Appendix B

Cosine rule

\[ a^2 = b^2 + c^2 - 2bc \cos \alpha \]  \hspace{1cm} (B.1)

Figure B.1  Cosine rule
Appendix B: Cosine rule


http://www.linuxdevices.com/articles/AT8906594941.html
http://www.tech9.net/rml/linux/
Index

accuracy, 11
ADAT, 29
ambient noise, 2
amplitude, 8
auto correlation function, 5
auto correlation function, 8
auto power spectrum, 10
bottleneck, 28
C++, 31
cardioid, 23
circle, 15
close microphone, 5
close miking, 2
complex conjugation, 6
convolution theorem, 7
correlation theorem, 6, 7
cross correlation function, 5
cross power spectrum, 6
delays, 15
digital signal processors, 1
Dirac delta pulse, 8
Direct Sound Source, 3
directional microphones, 2
discrete Fourier transforms, 6
distances, 15
division by zero, 10
Doppler, 42
dropouts, 21
filter, 8
Fourier transform, 6
Front Side Bus, 28
Hammerfall, 28
impulse response, 6
infra-red, 2
interpreted, 26
intersections, 16
latency, 31
lightpipe, 29
linear array, 16
Matlab, 25
mis-localization, 1
noise signals, 7
optical fiber, 29
parallelism, 26
patched, 31
phase, 8
POSIX threads, 31
power spectrum, 8
pre-whitening, 8

67
preempt, 31
prime factors, 41
public address, 1
Qt, 31
radius, 15
reflections, 10
relative height of the highest peak, 12
relative height method, 12
remote microphone, 5
remote miking, 2
resolution, 25
Roth processor, 9
S/PDIF, 29
sensitivity, 11
signal to noise ratio, 10
simulations, 3
single-threaded, 26
SMP, 26
source signal, 6
speech enhancement, 1
speech signal, 5
speed of sound, 5
stabilization factor, 10, 37
stage size, 12
stereo, 1
Tascam IF-TAD, 30
Tascam MA-AD8, 29
Tascam TM-D4000, 29
TDIF, 23
threshold value, 11
time lag, 5
Time Window Length, 12
time windows, 12
transfer function, 9
Trolltech, 31
unstable, 10
updates, 12
video, 2
wav, 25
wave-field synthesis system, 1, 12
white spectrum, 8
wireless, 2