Het toepassen van een microfoonarray op AMIGO

M.P.J. Craenmehr
m.p.j.craenmehr@student.tue.nl

Begeleider: ir. J. (Jos) Elfring

Afdeling Werktuigbouwkunde
Sectie Control Systems Technology
Technische Universiteit Eindhoven

8 september 2011

Samenvatting

Voor minder valide en oude mensen kunnen kleine taken al zeer vermoeiend zijn. Tot nu toe worden deze mensen geholpen door de thuiszorg, familieleden of vrienden. In de toekomst zal deze hulp echter ook kunnen komen van huishoudrobots.

AMIGO is een huishoudrobot ontwikkeld door TechUnited van de Technische Universiteit Eindhoven. AMIGO heeft een 3D-camera en een laser-scanner om zijn omgeving te zien. Aangezien de 3D-camera niet altijd alles in beeld kan hebben tijdens het helpen, is het handig om een microfoonarray te hebben die de omgeving afzoekt naar signalen. Met deze microfoonarray dient het mogelijk te zijn om de positie van de geluidsbron te achterhalen. Dit geeft AMIGO de mogelijkheid om naar de persoon te kijken en te handelen naar hetgeen dat wordt gevraagd. Tevens is het handig indien er meerdere personen in de kamer aanwezig te zijn. Met behulp van de microfoonarray kan dan achterhaald worden welke persoon er praat.

Voor het bepalen van deze positie in de 2D ruimte hoort het geluid opgenomen te worden door drie of meer microfoons op verschillende posities. Vervolgens worden er microfoonparen gevonden van de drie microfoons. De tijdsverschillen voor zachte signalen (fluiten en praten) worden berekend met behulp van de cross-correlatiefunctie. Deze functie bepaald het faseverschil tussen de twee signalen. Voor scherpe signalen (knippen met de vingers of tikken op tafel) worden de eerste korte pieken in het signaal vergeleken. Vervolgens wordt de hoek naar de geluidsbron bepaald met behulp van de geometrie van de microfoonarray. Deze stappen zijn allemaal opgenomen in een algoritme dat de signalen als invoer heeft en de hoek als uitvoer.

Voor de experimenten is een prototype van een microfoonarray van Sorama gebruikt. Sorama is een bedrijf binnen de TU/e dat de mogelijkheid biedt aan bedrijven om het geluid van hun apparaat in beeld te brengen. De microfoonarray bevat 64 MEMS (MicroElectroMechanical System) microfoons. Sorama heeft voor dit microfoonarray een circuit board ontwikkeld met FPGA (Field-Programmable Gate Array) en onboard geheugen. Dit biedt de mogelijkheid om het algoritme op de FPGA te flashen. De signalen uit de microfoons worden dan automatisch bewerkt door de FPGA. Indien het gehele algoritme op de FPGA staat, hoeft het board alleen de hoek naar de PC te sturen als er geluid aanwezig is. Het algoritme wordt in de *Soramadirectivitytool* (een tool ontwikkeld door Sorama om snel en makkelijk te meten) geïmplementeerd, zodat er met slechts één klik op de knop, eerst gemeten en dan de hoek tussen de geluidsbron en AMIGO wordt gegeven.

Het algoritme voor scherpe tonen is in staat om de hoek naar de geluidsbron te bepalen met een foutmarge van 11 graden (of 0.19 rad).

Het algoritme voor zachte tonen kan niet de juiste hoek naar de geluidsbron bepalen. De crosscorrelatie vindt wel de faseverschuiving die ook gevonden wordt als de gebruiker de signalen sterk uitvergroot. Deze faseverschuiving komt echter niet overeen met de verschuiving die verwacht wordt voor de hoek naar de geluidsbron.

Begrippenlijst

TDOA Time Difference Of Arrival

FFT Fast Fourier Transformation

SRP Steered Response Power

GCC Generalized Cross Correlation

PHAT PHAse Transform

SCOT Smoothed COherence Transform

AoA Angle of Arrival

MEMS MicroElectroMechanical System

FPGA Field-Programmable Gate Array

HDL Hardware Description Language

Inhoudsopgave

1 Introductie **5**

2 Literatuurstudie **6**
2.1 Verstoringen . . . 6
2.2 Van signalen naar tijdsverschil . . . 6
2.3 Van tijdsverschil naar positie . . . 9

3 Simulaties met MATLAB **11**
3.1 Opzet . . . 11
3.2 Resultaten . . . 12
3.3 Conclusie . . . 13

4 Algoritmes **14**
4.1 Verwerken signaal . . . 14
4.2 Output . . . 16

5 Experimenten **17**
5.1 Opstelling . . . 17
5.1.1 Verschillende afstanden van microfoonarray . . . 19
5.1.2 Reproduceerbaarheid metingen . . . 20
5.1.3 Resultaten zachte tonen . . . 21
5.2 Conclusie . . . 21

6 Conclusie en aanbevelingen **22**

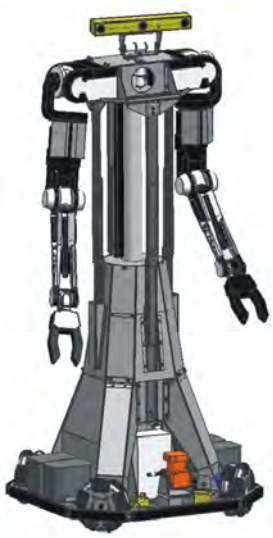
7 Bronnenlijst **23**

A Bijlagen **24**
A.1 Verschil tussen *far-field* en *near-field* . . . 24
A.2 De praktijk . . . 25
A.2.1 Hardware . . . 25
A.2.2 Software . . . 28
A.3 Vergelijken weegfuncties . . . 30
A.3.1 Opzet . . . 30
A.3.2 Resultaten . . . 31
A.3.3 Conclusie . . . 31
A.4 M-file voor scherpe signalen . . . 32
A.5 M-file voor zachte signalen . . . 36
A.5.1 Toewijzen ruimte in matrices . . . 36
A.5.2 Crosscorrelatie berekenen . . . 36
A.6 Experiment1 . . . 37
A.6.1 Opstelling . . . 37
A.6.2 Metingen . . . 37

1 Introductie

Voor minder valide en oudere mensen, kunnen alledaagse klusjes al veel moeite kosten. Het opstaan uit de stoel voor bijvoorbeeld het halen van een drankje kan voor deze mensen al zeer vermoeiend zijn. Er wordt dan ook volop gezocht naar het vergemakkelijken van deze klusjes.

De Technische Universiteit Eindhoven (TU/e) heeft de huishoudrobot AMIGO ontwikkeld om klusjes voor deze mensen te doen. AMIGO is uitgerust met een 3d-camera en een laser scanner om de omgeving in kaart te brengen. Tijdens dit werk, is AMIGO echter niet in staat om alle personen in beeld te houden. Het wordt voor de personen dan ook moeilijk om de aandacht van de robot te krijgen, zonder naar de robot toe te lopen.



Figuur 1: Huishoudrobot AMIGO ontwikkeld door CST van TU/e

Dit probleem kan opgelost worden door toepassing van een microfoonarray en nieuwe software. Met deze middelen is AMIGO in staat om de positie van de persoon te bepalen aan de hand van geluid. Ook is een microfoonarray in staat om de kwaliteit van het opgenomen geluid sterk te verbeteren. Zodoende zijn de functies van de microfoon (bijvoorbeeld spraakherkenning) sterk te verbeteren.

In dit verslag worden de ontwikkelingen rondom de microfoonarray voor AMIGO beschreven. Eerst zal gekeken worden naar de literatuur in hoofdstuk 2. Hier zal er onder andere gekeken worden naar de verschillende methoden voor bronlokalisatie. Vervolgens zal er in hoofdstuk A.2 gekeken worden naar de benodigdheden voor de implementatie van een microfoonarray. Zowel het hardware als software gedeelte zal hier behandeld worden. Hoofdstuk 4 beschrijft de algoritmes die zijn ontwikkeld, gevolgd door de resultaten van dit algoritme in hoofdstuk 5. Ten slotte wordt uit deze resultaten een conclusie getrokken en aanbevelingen gegeven voor eventueel opvolgende projecten.

2 Literatuurstudie

Voor het positioneren van een geluidsbron door middel van een microfoonarray, is het belangrijk om eerst de mogelijkheden te bekijken. In dit hoofdstuk staat de belangrijkste theorie beschreven die geraadpleegd is voor dit project.

2.1 Verstoringen

Het lokaliseren van de positie van de bron heeft enkele knelpunten. Een verkregen signaal via de microfoons is niet altijd bruikbaar. Eén van de factoren die de kwaliteit van het signaal kunnen verstoren is ruis. In een signaal zitten meerdere typen ruis. Ruis in een signaal kan meerdere oorzaken hebben:

Microfoon Zelfs bij het meten in een compleet stille ruimte zit er ruis in het signaal. Dit kan onder andere komen door de aanwezigheid van stroombronnen rondom de microfoon.

Omgeving De toepassing van de microfoonarray is in een rumoerige omgeving. Alle signalen behalve het nuttige signaal (dat geanalyseerd dient te worden) wordt dan gezien als ruis.

Omzetten analoog naar digitaal Bij het vervoeren van het signaal door kabels treed er ook ruis op. Deze ruis kan echter opgelost worden door het omzetten van het signaal van analoog naar digitaal zo dicht mogelijk bij de microfoons te doen. Of dit een mogelijkheid is, is afhankelijk van de gebruikte microfoonarray.

De kwaliteit van een signaal ten opzichte van de ruis wordt beoordeeld met de SNR-ratio. Deze *Signal to Noise Ratio* geeft de verhouding tussen het signaal en de ruis weer. Over het algemeen wordt een signaal met SNR >3 (of 0.5 dB) beschouwd als een bruikbaar signaal.

Behalve ruis zorgt ook reverberation voor verstoringen in het signaal. Reverberation is het weerkaatsen van geluid door de omgeving, die vervolgens opgevangen wordt door de microfoons. De invloed van reverberation op het signaal is dan ook sterk afhankelijk van de omgeving. Door reverberation bestaat de kans dat het geluid uit meerdere richtingen lijkt te komen, terwijl het maar één signaal is. De weerkaatsingen geven weer pieken in het signaal kort achter het echte signaal, zodat het lijkt dat er meerdere signalen zijn. Er zijn verschillende algoritmes ontwikkeld die het mogelijk maken om deze reverberation-pieken te herkennen of te verwijderen [1]. Later zal beschreven worden hoe er in dit project rekening is gehouden met reverberation.

2.2 Van signalen naar tijdsverschil

Voor de lokalisatie van de geluidsbron zijn er meerdere mogelijkheden. In dit verslag zal er alleen worden ingegaan op de methoden die gebruik maken van het verschil in aankomsttijden van geluid bij de microfoons: TDOA (Time Difference Of Arrival). In deze paragraaf zal er uitgewijd worden over de theorie om van de signalen van m verschillende microfoons, naar de TDOA te gaan.

Ten eerste wordt aangenomen dat het signaal dat de microfoon opvangt, beschreven kan worden met de volgende formule:

$$x_m(t) = h_m(t) * s(t) + b_m(t), \qquad m = 1, 2, ..., M \tag{1}$$

Hierbij is $x_m(t)$ het signaal dat opgevangen wordt en m het aantal microfoons. Behalve het signaal van de bron, $s(t)$, bevat dit signaal ook ruis $b_m(t)$. $h_m(t)$ is de impulsrespons (reactie van kamer op het geluid) tussen de microfoon en de geluidsbron.

Deze impulsrespons is te verdelen in het deel dat direct opgevangen wordt door de microfoon en het overgebleven deel $h'_m(t)$. Hiermee kan formule (1) omgeschreven worden naar formule (2). Hierbij is α_m de dempingsfactor van de microfoon op het signaal. tau_m is de vertraging van het signaal tot de microfoon.

$$x_m(t) = \alpha_m s(t - \tau_m) + h'_m(t) * s(t) + b_m(t), \qquad m = 1, 2, ..., M \tag{2}$$

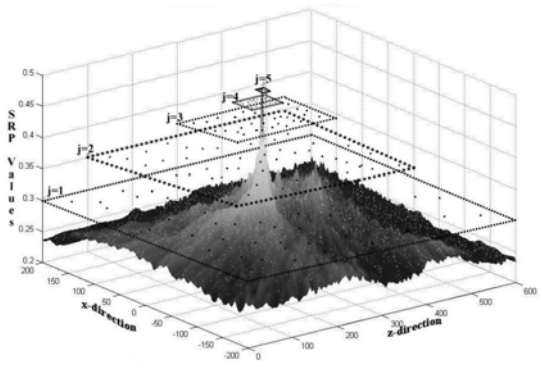
Dit signaal $x_m(t)$ wordt omgezet naar $X_m(\omega)$ in het frequentiedomein, door middel van een FFT (Fast Fourrier Transform) transformatie. Ten slotte worden er microfoonparen gekozen, waardoor het mogelijk wordt om het verschil tussen de twee ontvangen signalen te vinden. Deze microfoonparen kunnen vrij worden gekozen, zolang als ze maar onafhankelijk van elkaar zijn. In dit verslag wordt er ingegaan op twee methodes voor het vergelijken van de inkomende signalen: **SRP (Steered Response Power)** en **GCC (Generalized Cross Correlation)**.

SRP [2]: Bij dit principe wordt er over de beschikbare ruimte een raster van punten aangelegd. Dit raster kan over het algemeen vrij worden gekozen. Ten eerste kan men kiezen over welk gebied men de SRP wil analyseren. Voor dit project zou dit een vlak van bijvoorbeeld 10 bij 10 meter kunnen zijn. Vervolgens kan er gekozen worden op hoeveel punten men in dit vlak de SRP wil bepalen. Door meerdere punten aan te leggen duurt het berekenen langer, maar geeft het raster wel een beter beeld van de SRP over het oppervlak. Ten slotte kan men kiezen hoe groot het gebied van de volgende iteratie dient te zijn ten opzichte van het gebied van de vorige iteratie. Op de punten van het aangelegde rooster wordt vervolgens voor elk microfoonpaar de SRP berekend aan de hand van formule (3).

$$P(\mathbf{q}) = \int |\sum_{M=1}^{M} X_m(\omega) e^{j\omega\tau_m(\mathbf{q})}|^2 d\omega \tag{3}$$

De vector $\mathbf{q}$ bevat hierbij de positie van de punten op het raster. Rondom het punt met de maximale SRP, wordt een fijner raster aangelegd, waar weer het maximum van wordt bepaald. Zo wordt j iteraties een 2D positie gevonden. Deze methode kan ook gebruikt worden bij meerdere geluidsbronnen. Hierbij worden er nieuwe roosters aangelegd bij alle punten die een SRP hebben die hoger is dan een gekozen waarde. Vervolgens wordt bij al deze punten het maximum bepaald. Deze methode voor één geluidsbron is gevisualiseerd in figuur 2.

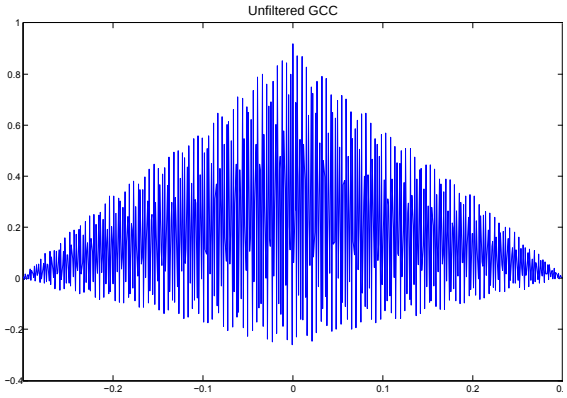
Een groot nadeel van deze methode is de kosten. Het duurt relatief lang (sterk afhankelijk van de hardware en het gekozen raster) om de SRP op de verschillende rasters te berekenen ten opzichte van andere methodes om de positie te bepalen. Dit maakt het moeilijk om deze methode in real-time applicaties toe te passen.



Figuur 2: Voorbeeld SRP bronlokalisatie.

Bij **GCC [3]** wordt er ook gebruik gemaakt van de faseverschuiving tussen beide signalen. Om deze faseverschuiving te bepalen wordt de cross-correlatiefunctie gebruikt. Deze functie staat weergegeven in formule (4). In figuur 3 staat een mogelijke uitkomst weergegeven voor de cross-correlatiefunctie.

$$R_m(\tau) = \int X_m(\omega) X^*_{m+1}(\omega) e^{j\omega\tau} d\omega \tag{4}$$



Figuur 3: Voorbeeld van cross-correlatiefunctie met faseverschuiving van 0.

Hierbij is $R_m(\tau)$ de correlatiefactor tussen signalen van microfoons m en m+1. $X_m(\omega)$ is het signaal $x_m(t)$ in het frequentiedomein. X^*_{m+1} is de complex geconjugeerde van microfoon m+1. De berekende correlatiefactor $R_m(\tau)$ is een functie van τ (tijdschaal). De τ waarvoor de correlatiefactor de hoogste waarde heeft, geeft de faseverschuiving aan tussen beide signalen.

Een groot probleem van deze methode is de brede piek die vaak gevonden wordt (zie figuur 3). In de literatuur zijn er meerdere weegfuncties te vinden, die in staat zijn om de piek voor de toepassing van dit project te versmallen. Door het signaal met deze weegfunctie te vermenigvuldigen worden de kleinere waarden verder gedempt. Voor real-time applicaties zijn er twee weegfuncties die veel gebruikt worden: GCC-PHAT (PHAse Transform) en GCC-SCOT (Smoothed COherence Transform). Aan het einde van de volgende paragraaf zullen met behulp van een opgesteld model deze weegfuncties vergeleken worden.

Met deze methode is het ook mogelijk om meerdere bronnen mee op te sporen. Hierbij zit echter wel een grote voorwaarde dat er van beide bronnen een smalle piek in de cross-correlatie wordt gevonden. Als dit niet het geval is overlappen de pieken elkaar, waardoor het niet duidelijk is hoeveel geluidsbronnen er zijn. Ook dienen de bronnen zich een aanzienlijke afstand van elkaar te bevinden, zodat de pieken van elkaar zijn te onderscheiden.

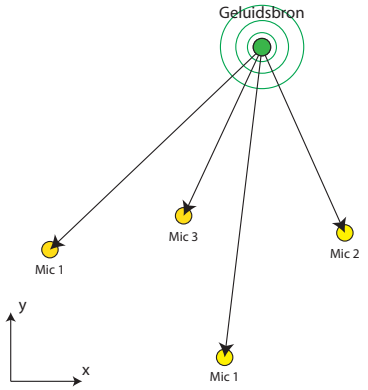
Door de vele mogelijkheden met GCC en de lage rekentijd zal er doorgegaan worden met deze methode.

2.3 Van tijdsverschil naar positie

Uit het verschil in aankomsttijden (TDOA), is uiteindelijk de positie van de geluidsbron te achterhalen. In deze paragraaf zullen er twee mogelijkheden worden beschreven om de relatieve positie van de geluidsbron uit de TDOA's te bepalen: 2D positie en AoA (Angle of Arrival).

2D positie: Met deze methode wordt er gebruik gemaakt van vereenvoudigde vergelijkingen voor bronlokalisatie. Hier wordt geen rekening gehouden met ruis in het signaal, zodat de positie precies te bepalen is.

We gaan uit van de volgende (ideale) situatie: Eén geluidsbron wordt opgenomen door 4 microfoons. De microfoons vangen geen ruis of weerkaatsingen van het signaal op. In figuur 4 is de situatie schematisch weergegeven.



Figuur 4: Het geluid van de bron wordt opgevangen door de microfoons.

De tijd die het geluid nodig heeft om van de geluidsbron tot de microfoons te komen, wordt gegeven door onderstaande vergelijking. c [m/s] is de snelheid van het geluid. i [-] geeft het nummer van de microfoon aan, terwijl de s aangeeft dat het over de geluidsbron gaat.

$$T_i = \frac{1}{c}\sqrt{(x_s - x_i)^2 + (y_s - y_i)^2} \qquad met : i = 1, 2, 3, 4 \tag{5}$$

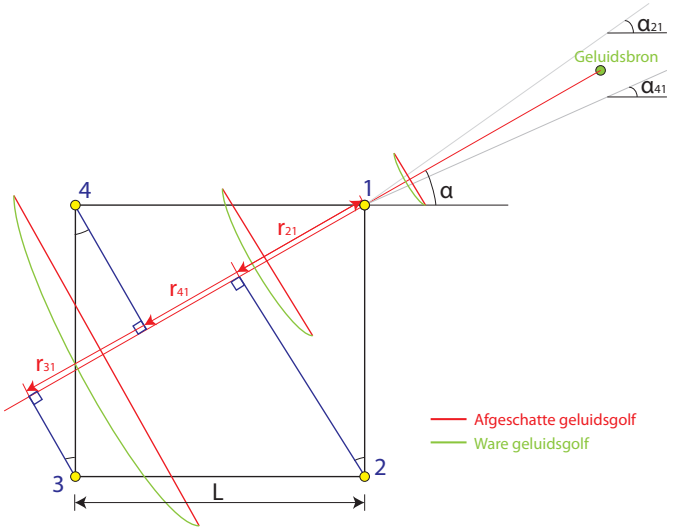
Hierbij zijn x_s [m] en y_s [m] (de positie van de geluidsbron) de enige onbekenden.

Door de positie van microfoon 1 op (0,0) te veronderstellen, kunnen de vergelijkingen omgeschreven worden naar het volgende:

$$dT_{i1} = T_i - T_1 = \frac{1}{c}(\sqrt{(x_s - x_i)^2 + (y_s - y_i)^2} - \sqrt{x_s^2 + y_s^2}) \qquad met : i = 2, 3, 4 \tag{6}$$

Bovenstaande vergelijking genereert drie hyperbolen. Het snijpunt van deze hyperbolen geeft de mogelijke locatie van de geluidsbron. In het volgende hoofdstuk staat de uitwerking van enkele simulaties met MATLAB om de 2D-positie te berekenen.

AoA (Angle of Arrival): Met behulp van de geometrie van de microfoons onderling is de hoek waaronder het geluid binnenvalt (AoA) te bepalen. In figuur 5 staat weergegeven hoe de geometrie eruit zou kunnen zien, indien er vier microfoons gebruikt zouden worden.



Figuur 5: Schematische weergave van positiebepaling middels 'Geometrie'.

In bovenstaand figuur is de geluidsbron aangegeven met een groene cirkel. Bij deze methode wordt de geluidsgolf als een vlak (rode lijnen) gezien i.p.v. een boog (groene lijnen). Dit is een juiste aanname voor far-field situaties, maar kan bij near-field situaties een probleem opleveren (zie bijlage A.1). Het verschil in de geluidsgolven is te zien in figuur 5. Door het afschatten van de geluidsgolf als vlak, wordt er niet één waarde gevonden voor α, maar één per microfoonpaar. Door de ΔT_{21} en ΔT_{41} van microfoon 1 te koppelen met de geometrie van de microfoonarray, zijn de hoeken α_{21} en α_{41} te bepalen met formule (7). Door deze twee hoeken te middelen, wordt uiteindelijk hoek α gevonden.

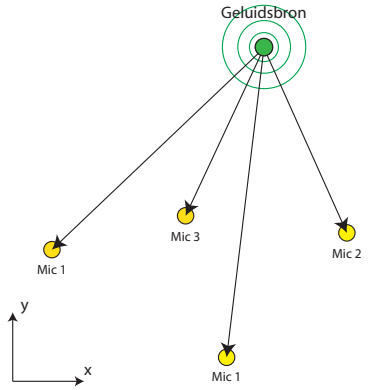
$$\alpha_{ij} = sin^{-1}(\frac{c \cdot \Delta T_{ij}}{L}), \qquad met: \Delta T_{ij} = \frac{r_{ij}}{c} \qquad met: i, j = 1, 2, 3, 4 \tag{7}$$

De fout die de hoeken α_{21} en α_{41} bevatten wordt groter naarmate de bron dichterbij de microfoonarray komt (*near-field*). Voor dit project is het echter vaak zo dat de persoon die roept, ver van de microfoonarray af staat. Verder is de microfoonarray dusdanig, dat de lengte L zeer klein is. Dit geeft ook de mogelijkheid om het punt waarvan de lijnen α_{21} en α_{41} vertrekken (in figuur 5) te verleggen naar het midden van de microfoonarray. Dit maakt het later makkelijker om de hoek te vertalen naar de hoek die AMIGO moet draaien.

Met behulp van dit model zijn er simulaties uitgevoerd om de weegfuncties PHAT en SCOT met elkaar te vergelijken. De uitwerking van deze simulaties zijn terug te vinden het volgende hoofdstuk.

3 Simulaties met MATLAB

Het kan soms nodig zijn om enkele simulaties in bijvoorbeeld MATLAB uit te voeren, om meer inzicht te krijgen in de theorie. Ook kan het nuttig zijn om een idee te krijgen van de rekentijd van een bepaalde functie. In dit hoofdstuk wordt uitgegaan van de volgende situatie: Eén geluidsbron wordt opgenomen door u microfoons (met u het aantal microfoons). In figuur 6 staat de situatie grafisch weergegeven.



Figuur 6: Het geluid van de bron wordt opgevangen door de microfoons.

3.1 Opzet

Zoals gegeven in hoofdstuk 2.3 zijn de formules voor de tijdsverschillen tussen de microfoonparen als volgt:

$$dT_{21} = T_2 - T_1 = \frac{1}{c}(\sqrt{(x_s - x_2)^2 + (x_s - y_2)^2} - \sqrt{x^2 + y^2}) \tag{8}$$

$$dT_{31} = T_3 - T_1 = \frac{1}{c}(\sqrt{(x_s - x_3)^2 + (x_s - y_3)^2} - \sqrt{x^2 + y^2}) \tag{9}$$

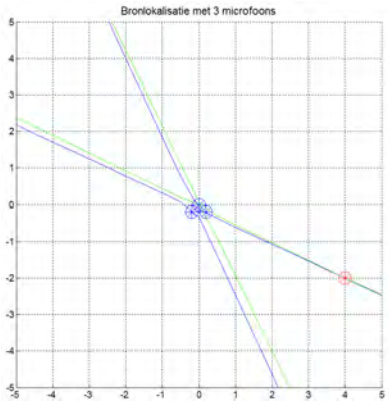
$$dT_{u1} = T_u - T_1 = \frac{1}{c}(\sqrt{(x_s - x_u)^2 + (x_s - y_u)^2} - \sqrt{x^2 + y^2}) \tag{10}$$

Hierbij is x_s de positie van de geluidsbron en is de positie van microfoon 1 gekozen op (0,0).Bovenstaande vergelijkingen genereren u-1 hyperbolen. Het snijpunten van deze hyperbolen geeft de locatie van de geluidsbron.

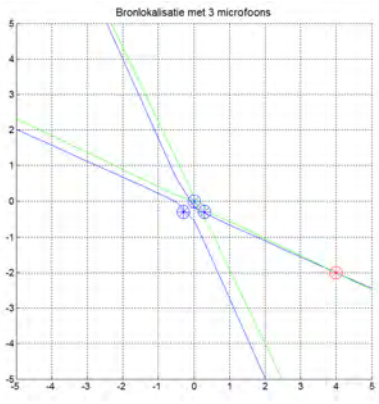
Deze vergelijkingen zijn in MATLAB gemplementeerd en er zijn simulaties met een verschillend aantal microfoons en posities van zowel microfoons als geluidsbron gedaan. Bij deze simulaties is eerst de geluidsbron vastgelegd op een bepaalde positie. Vervolgens berekent MATLAB de T_1, T_2, etc. uit. Met behulp van deze tijden worden dT_{21}, dT_{31}, etc. bepaald. Vervolgens wordt voor x_s een array opgesteld die van -5 [m] naar 5 [m] gaat, met stapjes van 0.1 [m]. Met deze array van x_s wordt vervolgens de mogelijk y_s berekent. De combinatie van deze x_s en y_s geven de mogelijke positie aan van de geluidsbron.

3.2 Resultaten

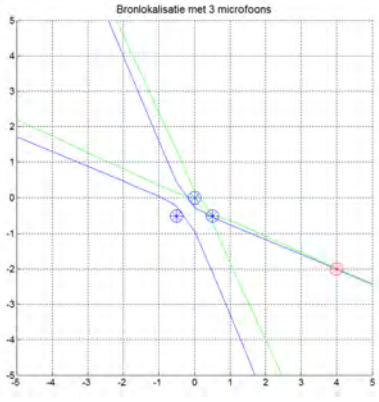
De resultaten van deze simulaties staan weergegeven in figuur 7(a), 7(b) en 7(c). Hierbij is de geluidsbron met een rode punt aangegeven. De posities van de microfoons zijn weergegeven als een blauwe punt.



(a) Positie microfoons: (0,0) , (-.2,-.2) en (.2,-.2)



(b) Positie microfoons: (0,0) , (-.3,-.3) en (.3,-.3)

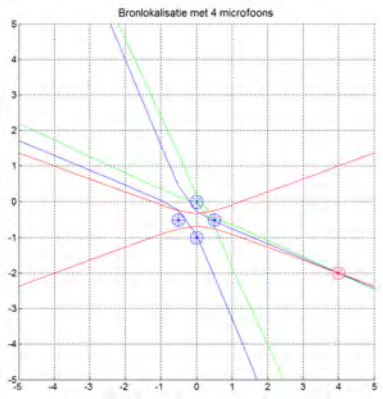


(c) Positie microfoons: (0,0) , (-.5,-.5) en (.5,-.5)

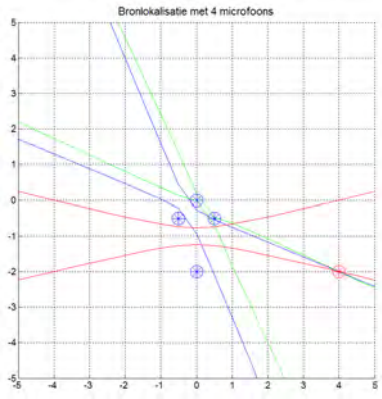
Figuur 7: Figuren van simulaties in MATLAB voor drie microfoons met geluidsbron op (4,-2)

Op de snijpunten van de groene en blauwe lijn zou dus mogelijk een geluidsbron kunnen liggen. In figuur 7(a) is goed het probleem te zien bij kleine microfoonarrays ten opzichte van de afstand tot de geluidsbron. De groene en blauwe lijn liggen zo dicht op elkaar, dat bij een kleine verstoring, het snijpunt ver verschuift. Dit kan gedeeltelijk opgelost worden, door de microfoons verder van elkaar te plaatsen. Hierdoor liggen de groene en blauwe lijn iets verder uit elkaar en is de verschuiving minder groot. Helaas is het niet altijd mogelijk om de microfoons ver uit elkaar te leggen. Verder hebben de groene en blauwe lijn meerdere snijpunten (rond (-2,5;5) ligt nog een snijpunt), waar de geluidsbron zich dus ook kan bevinden.

Om dit probleem op te lossen, kan men een microfoon toevoegen, waardoor er nog een lijn bijkomt. De simulaties hiervan staan weergegeven in figuur 8(a) en 8(b):



(a) Positie microfoons: (0,0) , (-.5,-.5), (.5,-.5) en (0,-1)

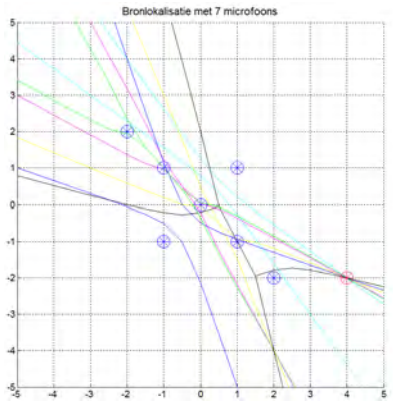


(b) Positie microfoons: (0,0) , (-.5,-.5), (.5,-.5) en (0,-2)

Figuur 8: Figuren van simulaties in MATLAB voor vier microfoons.

In figuur 8(b) is goed te zien dat het aantal mogelijke posities van de geluidsbron sterk afneemt. Er zijn nog maar twee posities waar de bron zich kan bevinden: In (4,-2) de goede positie, of rond het punt (1,-0.5). Door nog meer microfoons toe te voegen en de positie van de microfoons te veranderen, is het bepalen van de positie verder te optimaliseren. Aangezien dit niet het doel is van deze simulaties, is dit niet meer gedaan.

Ten slotte is er nog een simulatie gedaan met zeven microfoons, om te kijken hoe dit de resultaten en rekentijd zou benvloeden. Hierbij zijn twee microfoons iets verder uit het centrum geplaatst, om de lijnen iets uit elkaar te leggen. Het resultaat staat weergegeven in figuur 9.



Figuur 9: Positie microfoons: (0,0) , (1,-1) , (1,1) , (-1,1) , (-1,-1) , (2,-2) , (-2,2)

Hier is slechts één plaats waar alle 7 hyperbolen elkaar snijden, in de goede positie. De rekentijd neemt echter sterk toe voor het berekenen van de hyperbolen. Voor meerdere microfoons is dit dan ook een dure methode.

3.3 Conclusie

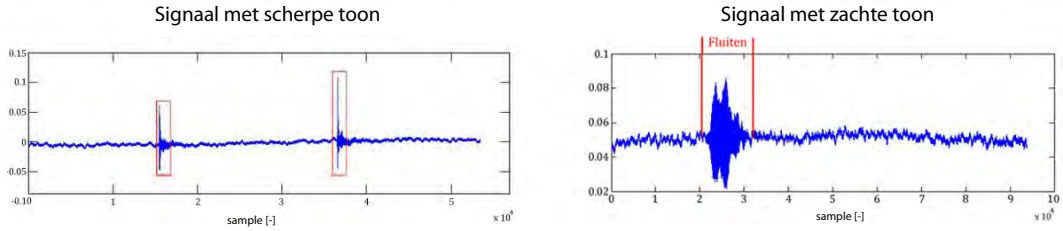
Het is met deze versimpelde vergelijkingen mogelijk een model op te stellen dat goed werkt voor een simulatie. Er moet echter bij vermeld worden dat voor het berekenen van de positie van de geluidsbron op deze manier ongeveer 3 seconden nodig is voor 3 hyperbolen. Verder is er bij de simulaties geen rekening gehouden met verstoringen. Aangezien het aanzienlijke tijd kost om de hyperbolen te berekenen, is er besloten om niet verder te simuleren met dit model.

4 Algoritmes

Het meten van het geluid geschiedt met de microfoonarray van Sorama. De onderdelen van deze microfoonarray en de gebruikte software voor het opbouwen en analyseren van het algoritme wordt uiteengezet in bijlage A.2. In dit hoofdstuk staat beschreven welke algoritmes er zijn ontwikkeld voor het lokaliseren van een geluidsbron.

4.1 Verwerken signaal

Tijdens het maken van het algoritme is er een onderscheid gemaakt tussen twee typen signalen: Scherpe en zachte tonen. Deze tonen zijn dusdanig verschillend, dat er ook verschillende algoritmes nodig zijn om de goede TDOA's te bepalen. Scherpe tonen bevatten scherpe pieken die snel weer uitdoven. Hierbij kan men denken aan het knippen van vingers of het tikken op tafel. Zachte tonen zijn vaak terug te zien in een groot deel van het signaal. Praten en fluiten zijn te beschouwen als zachte tonen. In figuur 10 staat een voorbeeld van beide signalen weergegeven. In dit hoofdstuk zullen voor beide signalen het gebruikte algoritme worden uiteengezet.

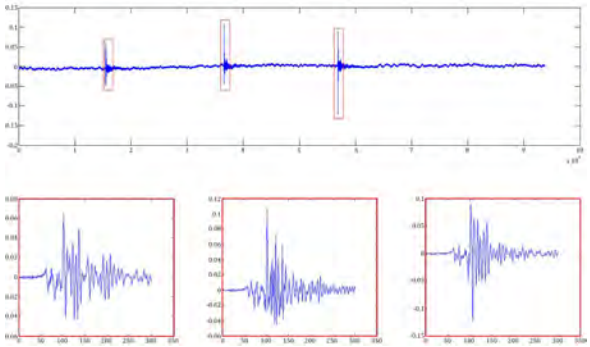


Figuur 10: Voorbeeld van signaal met scherp en zacht signaal.

Scherpe tonen

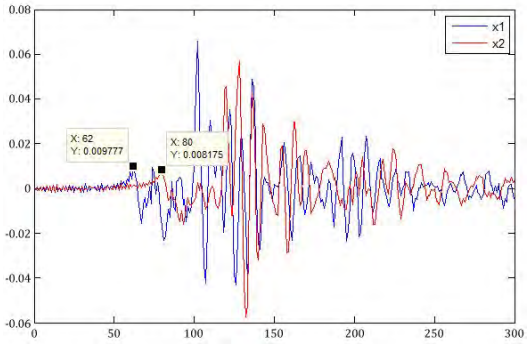
Om de aandacht van een persoon te krijgen, wordt behalve door te roepen ook vaak instinctief het knippen van de vingers gebruikt. Het menselijk oor is gevoelig voor korte scherpe tonen, waardoor dit vaak een succesvolle tactiek blijkt. Zoals te zien is in figuur 10 bevat slechts een klein deel van een dergelijk signaal nuttige informatie voor het bepalen van de positie. Aangezien we alleen deze pieken nodig hebben, herkent het algoritme pieken die hoger zijn dan een bepaalde waarde.

Vervolgens kan men instellen hoeveel samples men vóór en na de piek wil betrekken bij de analyse. Standaard staat dit op 300 samples, zodat er in plaats van het signaal slechts 300 samples overblijven voor elke piek. Om reverberation (weerkaatsingen) tegen te gaan, is het algoritme ingesteld dat na het opmerken van een piek, 2000 samples lang geen piek meer worden opgemerkt. Met de gebruikte samplefrequentie komt dit neer op 0.04 seconde. Deze tijd is dusdanig klein, dat twee losse signalen toch beide worden herkend, maar dat de reverberationpieken niet beschouwd worden als nieuwe signalen.



Figuur 11: Inkomend signaal met de gefilterde blokken.

Vervolgens worden deze blokken van verschillende microfoons met elkaar vergeleken. Ten eerste wordt de offset van het signaal afgehaald, door de waarde van het gemiddelde van de eerste 50 samples van het signaal af te trekken. Vervolgens zoekt het algoritme naar de eerste sample die een bepaalde waarde bereikt. In figuur 12 is te zien hoe dit eruitziet voor een voorbeeldsignaal.



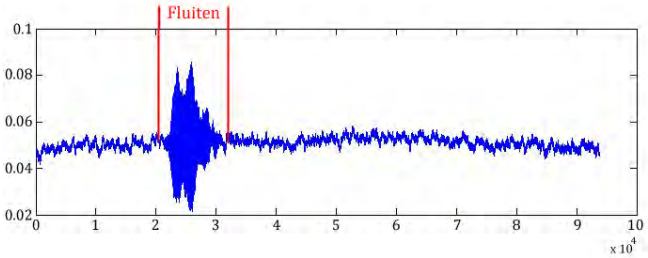
Figuur 12: Blok van voorbeeldsignaal van twee microfoons.

In deze figuur zijn de distinctieve toppen goed te zien. Deze top (en het eerste dal) wordt gebruikt om de verschuivingen van de signalen in samples te bepalen. Samen met de samplefrequentie geeft dit de TDOA per piek per microfoonpaar. Als laatste kan er gekozen worden om per piek in het signaal een hoek als output te hebben, of om al deze pieken te middelen en zo slechts één hoek te krijgen.

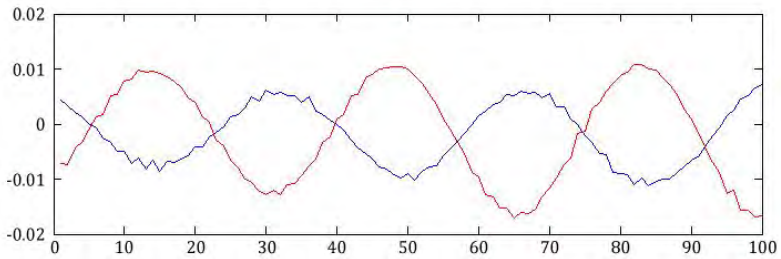
Zachte tonen

Een signaal met zachte tonen (fluiten, praten, etc.) bevat niet de korte, scherpe piek die wel te zien is bij scherpe tonen. Het beschreven algoritme zal dan ook voor deze signalen niet werken.

In figuur 13 staat een voorbeeldsignaal gegeven met een fluitsignaal. Door sterk in te zoomen op het fluitsignaal is te zien dat dit signaal te beschouwen is als twee sinussen met een faseverschuiving (te zien in figuur 14.



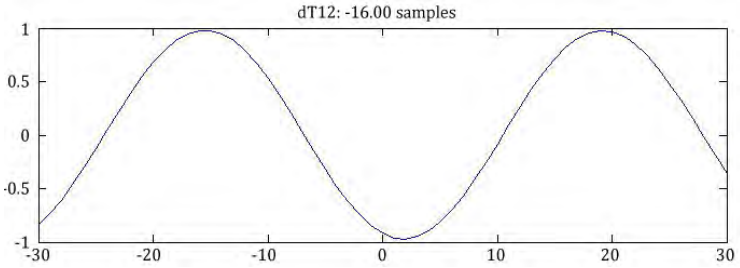
Figuur 13: Voorbeeld van een signaal met fluitsignaal



Figuur 14: Deel van voorbeeldsignaal sterk uitvergroot

Door de crosscorrelatie (faseverschuiving van het ene signaal ten opzichte van het andere signaal) van de twee signalen te berekenen, kan men de waarde bepalen bij elke tau. De waarde van tau waarbij de crosscorrelatie maximaal is, is de faseverschuiving tussen beide signalen.

Eén van de functies in MATLAB die de crosscorrelatie kan bepalen tussen twee signalen is '*xcorr*'. In figuur 15 staat de uitkomst van xcorr voor het voorbeeldsignaal. Het maximale sampleverschil tussen de microfoonparen is 0.20 [m] (diagonaal microfoonarray) gedeeld door 340 [m/s] (geluidssnelheid) maal 46875 [samples/s] (samplefrequentie). Dit komt neer op 28 samples. Vandaar dat alleen de -30 < tau < 30 wordt bekeken voor de crosscorrelatie.



Figuur 15: Deel van voorbeeldsignaal sterk uitvergroot

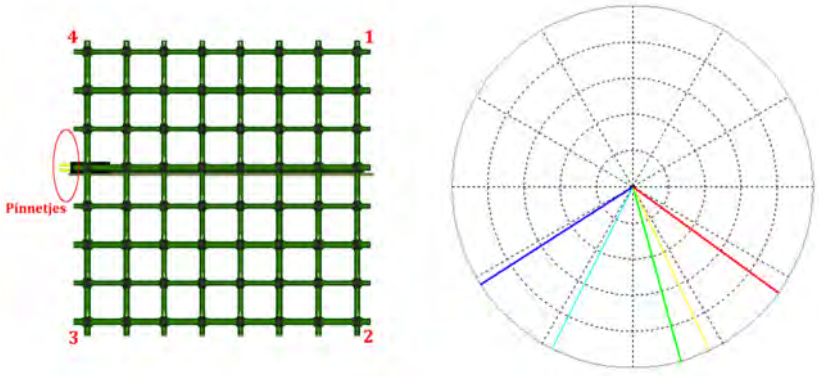
In bovenstaande figuur is te zien dat de crosscorrelatie twee toppen bevat. De maximale waarde van de croscorrelatie ligt bij -16 samples. Dit wil dus zeggen dat de verschuiving van het rode signaal 16 samples naar rechts voor dit voorbeeld het meest waarschijnlijk is. Het geluid komt dus 16 samples eerder aan bij microfoon 2, dan bij microfoon 1.

4.2 Output

Voor het bepalen van de positie van de bron is gebruik gemaakt van de methode die beschreven staat in hoofdstuk 2.3: AoA (Angle of Arrival). Met behulp van de geometrie van de microfoonarray worden twee microfoonparen opgesteld. De buren van de microfoon waar het signaal het eerst wordt opgemerkt vormen de microfoonparen. Voor bijvoorbeeld microfoon 2 zou dit 1-2 en 3-2 zijn.

De output van het algoritme is voor elke piek in het signaal voor elk microfoonpaar één verschiltijd. De twee verschiltijden worden voor elke piek gemiddeld, zodat er slechts een richting overblijft voor elke piek.

De richting van het signaal wordt vervolgens weergegeven in een '*polar plot*' van MATLAB. In figuur 16 staat een voorbeeld van een polaire plot gegeven. In deze figuur staat ook weergegeven hoe de output plot overeenkomt met de oriëntatie van de microfoonarray. De *pinnetjes* die zichtbaar zijn op microfoonarray wijzen naar links. Het algoritme dat beschreven is in dit hoofdstuk voor scherpe tonen staat in MATLAB-code in bijlage A.4.



Figuur 16: Output weergegeven in polaire plot (rechts) met bijbehorende oriëntatie microfoonarray (links)

5 Experimenten

Tijdens de ontwikkeling van de algoritmes ('Experiment 1') zijn er experimenten gedaan met de microfoonarray om de werking van het algoritme te testen. Verder zijn er na de ontwikkeling ('Experiment 2') nog metingen gedaan. In dit hoofdstuk worden alleen over 'Experiment 2' uitgebreid, aangezien 'Experiment 1' niet volledig betrouwbaar zijn. Dit experiment staat ter volledigheid toch weergegeven in bijlage A.6. Uit deze resultaten zal echter geen conclusie getrokken worden over de werking van de algoritmes.

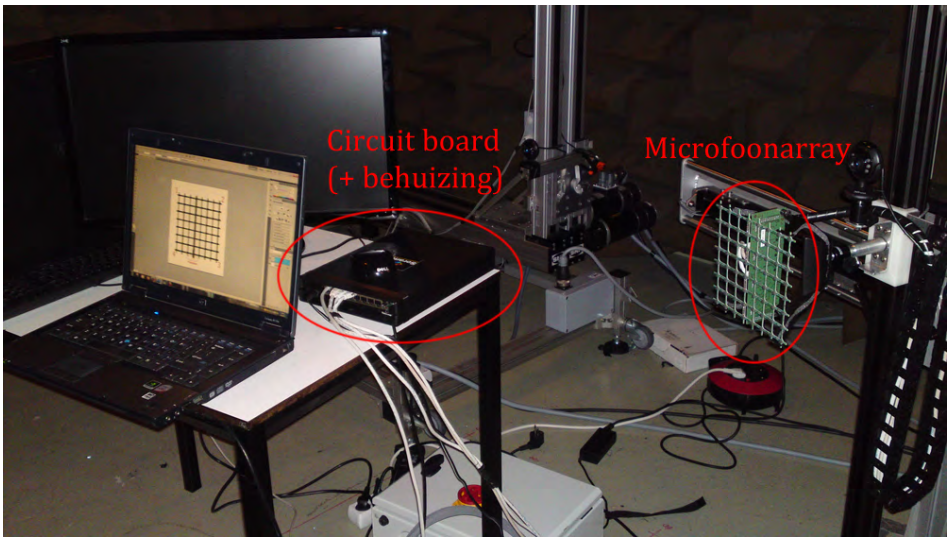
5.1 Opstelling

Tijdens het experimenteren stond de microfoonarray opgesteld in de geluidsvrije ruimte van Sorama. De wanden van deze ruimte zijn voorzien van een schuimlaag om de weerkaatsingen van het geluid te absorberen. Tevens hangt de kamer in een ophanging zodat trillingen van buitenaf niet worden opgemerkt in de kamer.

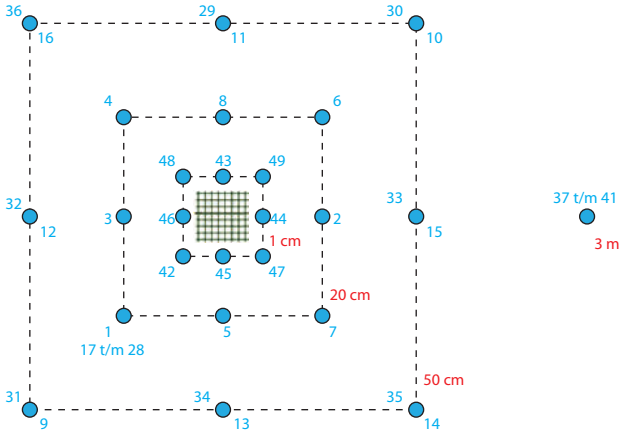
De microfoonarray stond op het moment van meten op de SoramaScan (de constructie die door Sorama wordt gebruikt voor metingen). De microfoonarray is via het circuit board verbonden met de laptop middels ethernet kabels. Op deze laptop wordt met de *Soramadirectivitytool* de meting gestart en de meetdata door het algoritme verwerkt.

Bij het experimenteren is er op meerdere punten rondom de microfoonarray gemeten. Er zijn op vier verschillende afstanden van de microfoonarray getest: 1 centimeter, 20 centimeter, 50 centimeter en 3 meter. Tevens is er op één positie twintig metingen verricht, om de reproduceerbaarheid van de resultaten te testen. Deze paragraaf is onderverdeeld in deze drie delen: 'Verschillende afstanden van microfoonarray', 'Reproduceerbaarheid metingen' en 'Zachte tonen'. De eerste twee delen zullen gaan over de experiment voor scherpe tonen. Het laatste deel gaat over de metingen voor het algoritme van de zachte tonen.

Deze opstelling staat weergegeven in figuur 17. In figuur 18 zijn alle meetpunten weergegeven met de lichtblauwe punten. Het nummer geeft het nummer van de meting aan, zoals de meting is opgenomen in de resultaten. In totaal zijn er 49 metingen verricht.



Figuur 17: Opstelling in geluidsvrije ruimte voor 'Experiment 2'



Figuur 18: Posities van de metingen ten opzichte van microfoonarray

5.1.1 Verschillende afstanden van microfoonarray

Voor elk meetpunt berekent het algoritme een hoek. Door deze waarde te vergelijken met de ingaande hoek (te bepalen middels de positie van het meetpunt), is de fout te bepalen. In tabel 1 staan de resultaten en de fout gegeven van de metingen op een vaste afstand van de microfoonarray. *kn* staat hierbij voor het knippen met de vingers, *kl* geeft aan dat er geklapt werd.

Tabel 1: Resultaten voor de metingen op verschillende afstanden van de microfoonarray

(a) Vingerknippen op 1 cm afstand

Meting	**Signaal**	**Hoek** [rad]	**Fout** [rad]
42	kn	3.92	0.01
43	kn	1.80	0.23
44	kn	0.30	0.30
45	kn	3.92	0.79
46	kn	3.16	0.02
47	kn	5.49	0.01
48	kn	2.36	0.00
49	kn	0.79	0.00
Gemiddelde fout:			**0.17 [rad]**

(b) Vingerknippen op 20 cm afstand

Meting	**Signaal**	**Hoek** [rad]	**Fout** [rad]
1	kn	3.79	0.14
2	kn	6.03	0.25
3	kn	3.19	0.05
4	kn	2.14	0.22
5	kn	4.94	0.23
6	kn	0.64	0.15
7	kn	5.73	0.23
8	kn	1.42	0.15
Gemiddelde fout:			**0.18 [rad]**

(c) Vingerknippen op 50 cm afstand

Meting	**Signaal**	**Hoek** [rad]	**Fout** [rad]
9	kn	3.79	0.20
10	kn	0.92	0.13
11	kn	1.43	0.14
12	kn	5.39	2.25
13	kn	4.72	0.01
14	kn	5.55	0.05
15	kn	0.01	0.01
16	kn	2.38	0.02
Gemiddelde fout:			**0.35 [rad]**

(d) Klappen op 50 cm afstand

Meting	**Signaal**	**Hoek** [rad]	**Fout** [rad]
29	kl	1.35	0.22
30	kl	0.79	0.00
31	kl	3.49	0.44
32	kl	3.12	0.02
33	kl	0.22	0.22
34	kl	4.97	0.26
35	kl	5.58	0.08
36	kl	2.36	0.00
Gemiddelde fout:			**0.16 [rad]**

Uit deze resultaten is af te leiden dat het algoritme werkt met een afwijking van ongeveer 0.20 rad. Hierbij dient ook opgemerkt worden dat meting 12 een uitschieter naar boven heeft van 2.25 rad. Toen dezelfde positie nogmaals werd getest kwam er een fout uit van 0.09 rad.

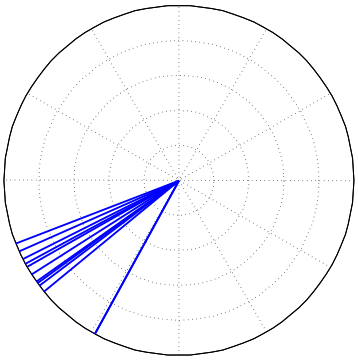
Verder is ook te zien dat de verhouding tussen de afstand ten opzichte van de microfoonarray en de fout, niet is zoals verwacht. Volgens de theorie van hoofdstuk 2.3 zal de fout namelijk kleiner worden, als de afstand tussen de microfoonarray en de geluidsbron groter wordt. Eén mogelijke verklaring, is dat de SNR groter wordt, door de grotere afstand die het geluid aflegt. Om dit te bevestigen zouden er echter eerst meer metingen moeten worden gedaan.

5.1.2 Reproduceerbaarheid metingen

Voor de reproduceerbaarheid van de opstelling te testen, zijn er meerdere metingen op dezelfde positie gedaan ten opzichte van de microfoonarray. De resultaten van deze metingen staan weergegeven in tabel 5.1.2. In figuur 19 staan de berekende hoeken weergegeven met de blauwe lijnen.

Tabel 2: Resultaten van de metingen op een vaste positie ten opzichte van microfoonarray

Meting	**Signaal**	**Hoek** [rad]
17	kn	4.21
18	kn	3.61
19	kn	3.56
20	kn	3.79
21	kn	3.83
22	kn	3.76
23	kn	3.51
24	kn	4.21
25	kn	3.77
26	kn	3.66
27	kn	3.64
28	kn	3.71
Gemiddelde hoek:		**3.77 [rad]**
Standaardafwijking:		**0.22 [rad]**



Figuur 19: Polaire plot van de berekende hoek (blauwe lijn) en ingaande hoek (groene lijn)

In figuur 19 en tabel 5.1.2 is te zien dat de meeste metingen dicht bij elkaar liggen (binnen 0.32 rad of 19 graden). Er zijn echter twee metingen (Meting 17 en 24) die buiten deze grenzen vallen. Deze twee metingen verschillen ongeveer 0.53 rad of 30 graden van de andere metingen.

De standaardafwijking van de meetresultaten is, berekend met formule 11, 0.216 rad.

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}, \qquad met: \mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{11}$$

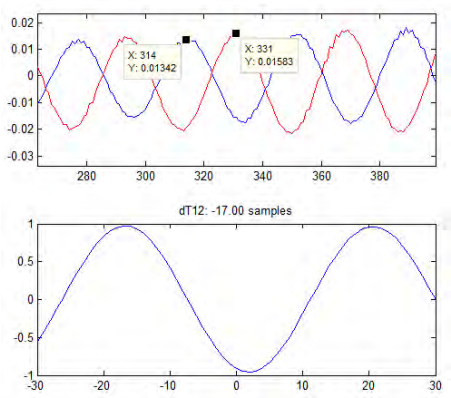
5.1.3 Resultaten zachte tonen

Voor de zachte tonen zijn er metingen gedaan onder twee hoeken. De resultaten staan weergegeven in tabel 3. Hierbij is de ingegeven hoek de hoek waaronder het geluid is ingevoerd in de microfoon. Dit zou dan in het ideale geval ook overeenkomen met de gevonden hoek.

Tabel 3: Berekende hoeken uit metingen zachte tonen

Meting:	**Ingegeven hoek [rad]**	**Berekende hoek [rad]**
Zacht1	-2.36	1.58
Zacht2	-0.79	1.58

Uit bovenstaande resultaten is te zien dat deze niet overeenkomen. Dit komt door een fout bij het binnenhalen van het signaal. Door de signalen van de microfoons sterk uit te vergroten, kan men zelf de faseverschuiving tussen de signalen afschatten. Tijdens het ontwikkelen van het algoritme is dit vaker gedaan, waarbij men tot de conclusie kwam, dat het algoritme wel de goede faseverschuiving bepaald tussen de signalen. In figuur 20 is bijvoorbeeld te zien dat de gevonden faseverschuiving van 17 samples klopt. Deze faseverschuiving komt echter totaal niet overeen met de faseverschuiving die verwacht wordt onder deze hoek (namelijk rond de 12 samples).



Figuur 20: Bepalen tijdsverschil zacht signaal

5.2 Conclusie

Zoals te zien in tabel 1 ligt de gemiddelde fout van het algoritme op 0.20 rad of 11.5 graden. Verder is er in figuur 19 te zien dat de berekende hoek van verschillende metingen op de zelfde positie met elkaar overeenkomen. Op twee uitschieters na, liggen de tien metingen binnen 0.32 van elkaar. Met deze resultaten is de werking van dit algoritme aangetoond.

Voor het algoritme van de zachte tonen is aangetoond dat deze niet werkt. Het algoritme is wel in staat om het faseverschil te vinden, maar niet in staat om uit deze faseverschillen de hoek naar de geluidsbron te bepalen.

6 Conclusie en aanbevelingen

De microfoonarray van Sorama is goed in staat om als microfoonarray gebruikt te worden voor AMIGO. Aangezien op dit moment slechts vier van de 64 microfoons worden gebruikt, is het eventueel ook mogelijk om een microfoonarray door Sorama te laten maken met minder microfoons. Dit heeft ook als grote voordeel dat de microfoonarray op maat gemaakt kan worden voor AMIGO met flexibele flexprints.

De integratie van de microfoonarray met de mini-pc's op AMIGO zou goed moeten gaan. Zowel de chip die de USB-verbinding regelt, als de chip die de PCI-e-verbinding mogelijk maakt, zijn wereldwijd in gebruik en hebben voor Windows, Linux en Mac OS ondersteuning.

Het algoritme voor scherpe tonen is in staat om (met een foutmarge van 11 graden) de hoek naar de geluidsbron te bepalen. Voor zwakke tonen (praten, fluiten) werkt het algoritme nog niet naar behoren. De correlatiefunctie vindt het faseverschil dat men ook vindt als men het signaal sterk uitvergroot, maar dit faseverschil geeft niet de juiste hoek naar de geluidsbron.

De *Soramadirectivitytool* is op dit moment niet in staat om real-time metingen te verrichten. Door de metingen met deze tool snel achter elkaar te laten geschieden, wordt real-time nagebootst. Voor deze toepassing zou dit genoeg kunnen zijn, om bijvoorbeeld iedere halve seconden het signaal te analyseren en alleen de hoek uit te schrijven naar AMIGO.

Een volgende stap zou het integreren van de goede algoritmes op de FPGA van het circuit board kunnen zijn. Hierdoor wordt de computer ontlast van al het rekenwerk. Tevens kan een FPGA dit type berekeningen effectiever afhandelen dan de CPU van een PC. Xilinx ISE Design Suite in combinatie met MATLAB biedt een goede mogelijkheid om een dergelijk schema op te bouwen met de Xilinx blockset. Het is wel aan te raden om dit door iemand te laten doen die ervaring heeft met het flashen van FPGA's, aangezien de onderlinge verbindingen van de blokken goed moet zijn. Als de FPGA geflashed is, hoeft de computer alleen de richting van het geluid uit te sturen naar AMIGO door USB of PCI-e.

7 Bronnenlijst

Referenties

[1] Adaptive Digital Technologies; *Echo Cancellation algorithms, echo cancellation / vqe software.* August 31, 2011.

[2] Weiping Cai, Shikui Wang, Zhenyang Wu; *Accelerated steered response power method for sound source localization using orthogonal linear array.* August 27, 2009.

[3] Joshi Kunal; *Real Time2-D Audio Source Localization on TIs 320C6713DSK (in embedded C)* March 13, 2011.

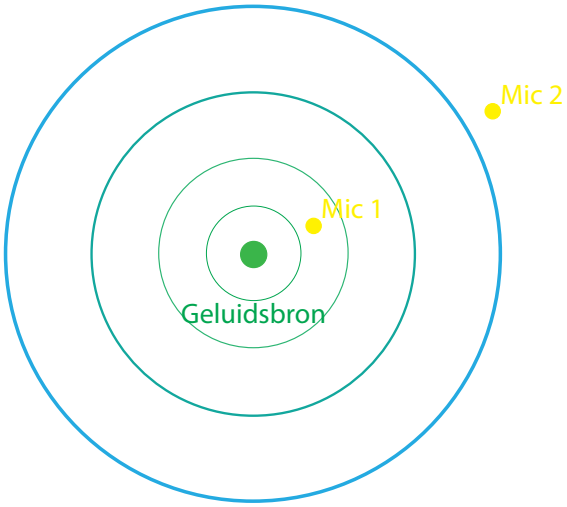
[4] http://www.earsandgears.com; *Microphone Directionality - Polar Patterns.* October 28, 2010.

[5] http://www.ftdichip.com; *Drivers.* August 18, 2011.

A Bijlagen

A.1 Verschil tussen *far-field* en *near-field*

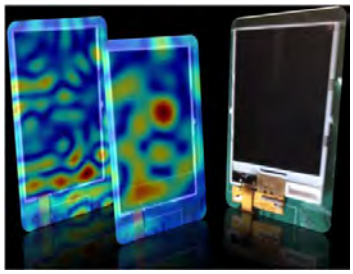
De vorm van de geluidsgolf is afhankelijk van de afstand van de geluidsbron tot de microfoonarray. Als deze afstand klein is, wordt er gesproken over *near-field*. De geluidsgolf wordt in deze situatie beschouwd als een bol (3D) of cirkel (2D). Als de afstand tussen de geluidsbron en microfoonarray groot is, wordt de situatie omschreven als *far-field*. Hier kunnen de geluidsgolven als een vlak (3D) of rechte lijn (2D) worden beschouwd.



Figuur 21: Geluidsbron met Mic 1 als *near-field* microfoon en Mic 2 als *far-field* microfoon.

A.2 De praktijk

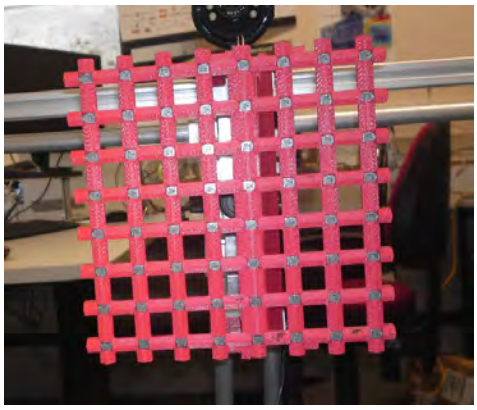
Voor dit project is een prototype van de microfoonarray van Sorama gebruikt. Sorama is een bedrijf dat de mogelijkheid biedt aan bedrijven om geluid in beeld te brengen. Door met een microfoonarray het apparaat of onderdeel te scannen, kan men op de millimeter nauwkeurig bepalen waar het geluid van af komt. Vervolgens stelt Sorama enkele oplossingen voor aan de klant om de akoestiek van het apparaat te verbeteren. Dit zorgt uiteindelijk voor een beter eindproduct van het bedrijf.



Figuur 22: Output van geluidsanalyse op scherm gsm

A.2.1 Hardware

Sorama beschikt over meerdere microfoonarrays. Voor dit project is er gebruik gemaakt van het nieuwste prototype. Bij deze microfoonarray zit een speciaal door Sorama ontwikkeld circuit board, dat het doorsturen en verwerken van de data op zich neemt. In dit hoofdstuk zullen de belangrijkste onderdelen van de microfoonarray en circuit board worden uiteengezet.



Figuur 23: De microfoonarray van Sorama

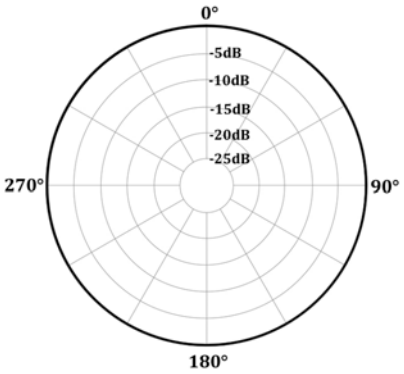
Microfoonarray

De microfoonarray bestaat uit een matrix van 8x8 microfoons. Deze microfoons zijn op een afstand van twee centimeter van elkaar geplaatst, wat de totale grootte van de microfoonarray op 14 bij 14 cm brengt. De microfoons zijn MEMS (MicroElectroMechanical System) microfoons. Een MEMS microfoon bestaat uit een dun plaatje silicone dat het membraan vormt. Dit membraan wordt door de trillingen in beweging gebracht. De inwendige chip versterkt dit analoge signaal en zet dit om naar een digitaal signaal. Door het signaal dicht bij het membraan om te zetten, wordt de ruis die geïntroduceerd wordt door de omgeving geminimaliseerd. In figuur 24 staat een MEMS microfoon weergegeven. Rechtsboven is het membraan (witte schijfje) goed te zien.

Figuur 24: MEMS microfoon

Het grote voordeel van deze microfoon is de digitale output, de beperkte grootte en de directionaliteit. De digitale output zorgt voor minder kwaliteitsverliezen tijdens het overbrengen van het signaal. Ook zorgt dit ervoor dat er geen versterker nodig is achter de microfoons. De microfoons kunnen op zeer kleine schaal worden gemaakt, door de technologische vooruitgang bij het produceren van micro-elektronica.

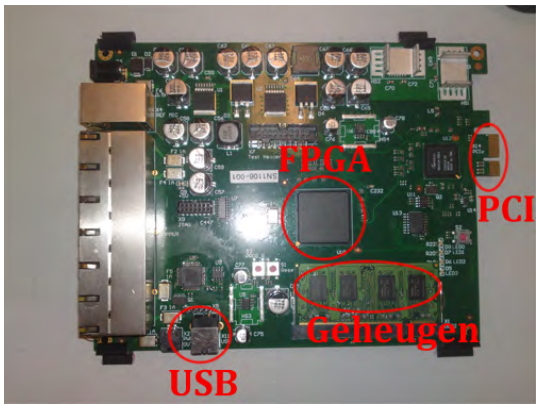
Ten slotte zijn deze microfoons omnidirectionaal. Een omnidirectionele microfoon heeft in het ideale geval een perfecte bolvormige karakteristiek. Dit betekent dat de microfoon even gevoelig is voor geluid uit alle richtingen. Dit polaire patroon staat weergegeven in figuur 25. Het kan echter zo zijn dat het lichaam van de microfoon zelf geluid 'in de weg' zit, waardoor geluid uit deze hoek wegvalt. Zeker voor hoge frequenties (een korte golflengte) is het toepassen van deze kleine microfoons een voordeel [4].



Figuur 25: Het polaire patroon van een omnidirectionele microfoon

Het circuit board

Het circuit board (zie figuur 26) bevat vele onderdelen die het samen mogelijk maken om het signaal van de microfoons door te sturen en eventueel te bewerken. In dit deel zullen de belangrijke onderdelen van dit board kort worden aangehaald.



Figuur 26: Circuit board van microfoonarray Sorama

Eén van de onderdelen is de Xilinx® **FPGA (Field-Programmable Gate Array**. Deze Programmable logic device (PLD) is met behulp van logische functies in staat schakelingen en zelfs (eenvoudige) wiskundige functies te vormen. Xilinx is één van de marktleiders op het gebied van de productie van PLD's. Behalve de productie van de chips, neemt het ook een deel van de software op zich. Door de toenemende complexiteit van deze chips, is HDL (Hardware Description Language) nodig om het systeem te definiëren. Met behulp van de Xilinx ISE Design Suite is het mogelijk om schema's te ontwerpen die met dezelfde Design Suite direct zijn om te zetten naar HDL-code. Bij de Xilinx ISE Design Suite wordt ook de *Xilinx Blockset* geleverd. Deze blockset is te gebruiken in Simulink (platform binnen MATLAB). Dit geeft de mogelijkheid om in Simulink een schakeling te ontwerpen met deze 'Xilinx Blockset'. Vervolgens wordt deze schakeling door de Xilinx ISE Design Suite omgezet naar HDL-code. De gevormde HDL-code kan vervolgens naar de FPGA geflashed worden.

Het door de FPGA bewerkte geluidssignaal kan vervolgens op twee manieren naar de PC worden doorgezet: **USB** of **PCI**. De chip die verantwoordelijk is voor het verzenden van de data over het USB-kanaal, is van FTDI (Future Technology Devices International Ltd.). Een groot voordeel van de chip, is dat er meerdere platforms ondersteund worden, namelijk Windows, Linux en MAC OS [5]. Voor de PCI verbinding wordt de *GN4124 PCI Express Bridge* gebruikt. Deze chip van Gennum® is speciaal ontwikkeld om samen te werken met FPGA's.

Aangezien er **geheugen** op het circuit board zit, zou een groot deel van het verwerken van het signaal gebeuren op de microfoonarray zelf. Dit ontlast voor een groot deel de computer. Een bijkomend voordeel is dat voor het verwerken van geluidssignalen, een FPGA effectiever werkt dan de CPU die in de PC aanwezig is. De CPU voert namelijk slechts de instructies uit die ingegeven worden. De FPGA is geflashed, zodat er de logische poorten ook daadwerkelijk zijn gevormd. Dit zorgt ervoor dat het signaal veel sneller wordt verwerkt.

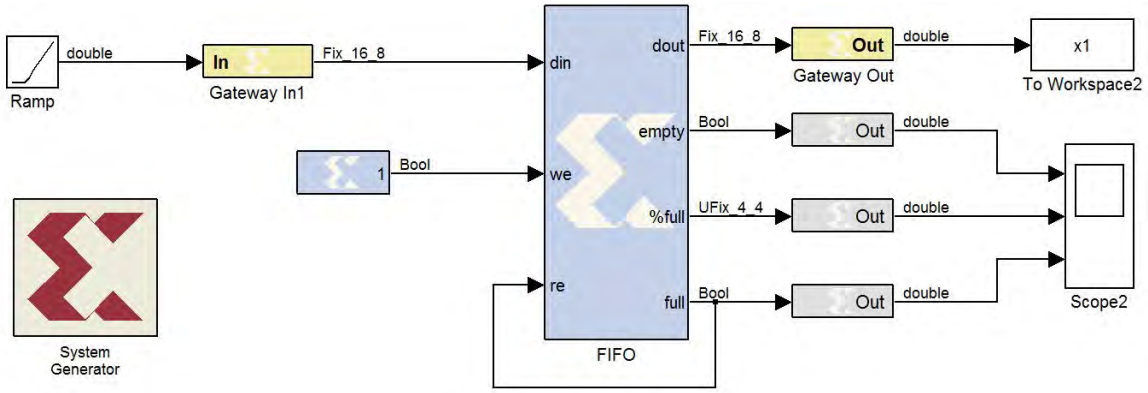
A.2.2 Software

Voor het omzetten van het algoritme naar de FPGA is zoals eerder gezegd de Xilinx ISE Design Suite nodig. Verder is er ook gebruik gemaakt van het platform Simulink in MATLAB. In dit deel zal er verder worden ingegaan op deze software.

MATLAB® (Simulink)

MATLAB is een programma uitgegeven door The Mathworks, dat in de technische wereld veel wordt gebruikt om problemen mee door te rekenen. In Simulink (een platform binnen MATLAB) is het mogelijk om grafisch schakelingen te modelleren met behulp van afzonderlijke blokken. De gemodelleerde schakelingen kunnen vervolgens gesimuleerd, geanalyseerd en geoptimaliseerd worden in de MATLAB-omgeving.

Voor het overschrijven van de Xilinx FPGA chip, dient er in Simulink de *Xilinx blockset* gebruikt te worden. Deze blockset (meegeleverd met Xilinx ISE Design Suite) bevat alle blokken die door de Xilinx ISE Design Suite omgezet kunnen worden naar HDL-code. In figuur 27 staat een voorbeeld gegeven van een schakeling met Xilinx blokken.



Figuur 27: Voorbeeld van schakeling in Simulink met Xilinx blockset.

In figuur 27 zijn drie cruciale blokken te zien van de Xilinx blockset, die in elke schakeling van Simulink aanwezig dienen te zijn: **Gateway In**, **Gateway Out** en **System Generator**.

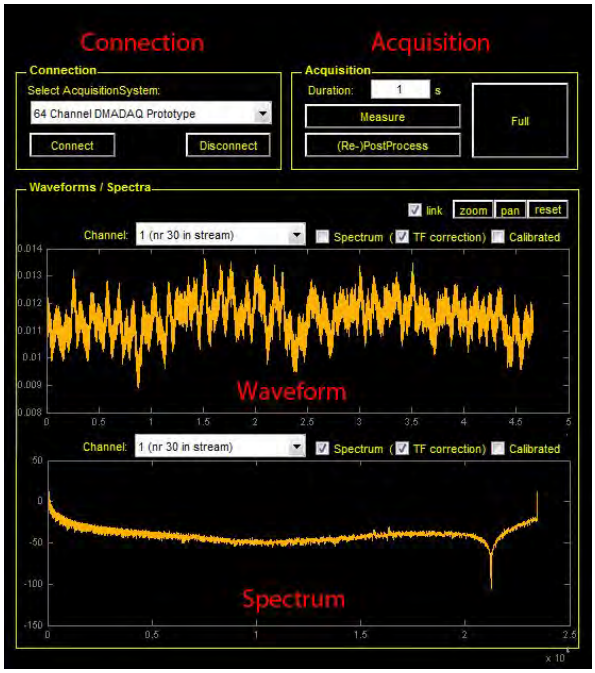
De *Gateway In/Out* geven aan welk deel van de schakeling beschouwd worden alsof het op de FPGA staat. De blokken die tussen de Gateways staan, dienen dan ook Xilinx blokken te zijn.

De *System Generator* geeft de instellingen van de simulaties. Hier kan bijvoorbeeld het type FPGA ingegeven worden. Ook kan er de manier en plaats van het wegschrijven van de HDL-code bepaald worden. Met dit blok kan er ook uiteindelijk voor worden gekozen om de HDL-code te genereren.

Met bovenstaande blokken en de andere blokken (in totaal meer dan 90 blokken) van de Xilinx Blockset zijn de gewenste schakelingen te vormen. Hierbij dient men er echter rekening mee te houden dat het maken van schakelingen in Xilinx geheel anders is dan een normale schakeling in Simulink. Aangezien het hier gaat om het programmeren van hardware, moet er veel meer informatie worden opgegeven dan bij een normale Simulink schakeling. Hierbij dient men bijvoorbeeld te denken aan de de significantie van de getallen, de onderlinge timing van de blokken en dient er rekening gehouden te worden met (afrond)fouten.

Soramadirectivitytool

Sorama heeft voor het opnemen van geluid en het wegschrijven van data naar de PC, speciale software ontwikkeld. Verder is er mede voor dit project een GUI (Guided User Interface) ontwikkeld om het mogelijk te maken met slechts één klik te meten en de data naar MATLAB te sturen: '*Soramadirectivitytool*'. Deze tool geeft ook de signalen weer van alle 64 microfoons. In figuur 28 staat de lay-out van de *Soramadirectivitytool* weergegeven.



Figuur 28: Layout Soramadirectivitytool.

Connection: Hier kan men het type van de microfoonarray instellen. Bij dit project is er gebruik gemaakt van '64 Channel DMADAQ Prototype'. Dit is de microfoonarray met 64 microfoons die te zien is in figuur 23.

Acquisition: Voor het instellen van de tijdsduur van de meting. Verder kan men hier kiezen voor alleen meten, het nabewerken van het laatste signaal of een combinatie van beiden.

Waveform: Deze grafiek geeft het laatst opgenomen signaal weer als op *(Re-)PostProcess* wordt geklikt. Bij *Channel* kan er gekozen worden voor kanaal 1-64 of alle 64 kanalen tegelijk. Verder kan er gekozen worden om het signaal gekalibreerd (aan de hand van kalibratiebestand) of niet-gekalibreerd te weergeven. Verder is er de mogelijkheid om in te zoomen op een gewenst deel van het signaal met de optie *zoom*.

Spectrum: Hier wordt het spectrum van het gekozen kanaal weergegeven. Ook hier kan men inzoomen om een duidelijker beeld van het spectrum te verkrijgen.

Het gemeten signaal wordt bewerkt indien er op *(Re-)PostProcess* wordt geklikt. Vóór het bewerken van het signaal dient er gekozen te zijn voor het algoritme dat de positie van de bron uit het signaal haalt. Het gewenste algoritme wordt in de M-file van de *Soramadirectivitytool* geplaatst. Dit algoritme wordt in hoofdstuk 4.1 uiteengezet.

A.3 Vergelijken weegfuncties

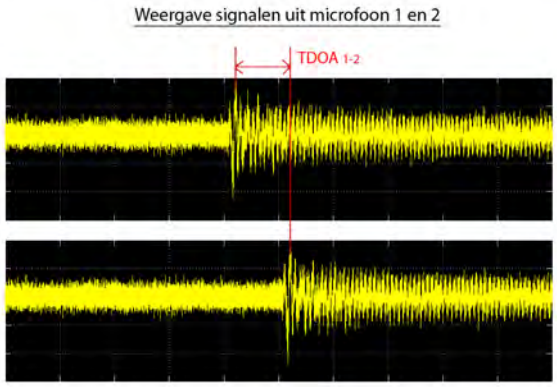
Voor de GCC methode zijn er twee weegfuncties beschikbaar die ongeveer dezelfde uitkomst hebben: GCC-PHAT en GCC-SCOT. Deze weegfuncties worden gebruikt om de piek van de correlatiefunctie te versmallen. Dit geeft uiteindelijk een betere afschatting van de tijdsverschillen en dus positie van de bron. In formule (12) en (13) staan de weegfuncties gegeven.

$$W^{GCC} = \frac{1}{|X_0(f)||X_1(f)|} \tag{12}$$

$$W^{SCOT} = \frac{1}{\sqrt{X_0(f) \cdot X_0^*(f) \cdot X_1(f) \cdot X_1^*(f)}} \tag{13}$$

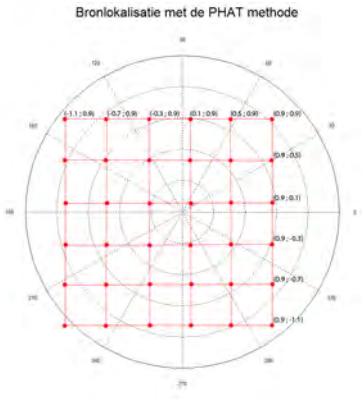
A.3.1 Opzet

Ten eerste is er in Simulink een model ontworpen, dat vier signalen simuleert, met verschillende vertragingen. Deze vertragingen simuleren de tijdsverschillen tussen de microfoons. De vier signalen bestaan uit een geluidsgolf die n keer wordt gesimuleerd. Vervolgens wordt bij dit signaal een willekeurige ruis opgeteld, om de ruis van de omgeving na te bootsen. In figuur 29 staan deze signalen weergegeven. Hierbij zijn de vertragingen echter sterk uitvergroot.



Figuur 29: Twee van de vier signalen gebruikt voor de simulatie.

De vertragingen die gebruikt worden voor de signalen, worden door een m-file uitgerekend uit de opgegeven positie van de bron. Deze m-file is ook in staat om uit de signalen de hoek naar de geluidsbron te bepalen met het AoA-model. Door in deze m-file de weegfunctie te veranderen, kan gekeken worden welke weegfunctie de kleinste error geeft voor bepaalde posities van de geluidsbron. In figuur 30 staat het raster met de gesimuleerde positie van de geluidsbron weergegeven.



Figuur 30: Het raster geeft gesimuleerde posities van de geluidsbron weer.

A.3.2 Resultaten

Met deze opzet is het mogelijk om voor elk punt op het raster, de hoek te berekenen naar de geluidsbron. Aangezien de positie van de geluidsbron zelf wordt ingevoerd, is de werkelijke hoek ook bekend. Door deze twee hoeken te vergelijken, wordt de fout bekend. Doordat de condities voor alle tests hetzelfde zijn, is het mogelijk om op deze manier GCC-PHAT met GCC-SCOT te vergelijken.

GCC-PHAT:

Tabel 4: Fout in rad tussen de berekende en werkelijke hoek voor PHAT

	-1.1	**-0.7**	**-0.3**	**0.1**	**0.5**	**0.9**
-1.1	2.8e-3	2.2e-2	3.2e-2	9.1e-2	1.9e-2	9.4e-3
-0.7	1.8e-2	2.8e-3	2.8e-3	5.6e-2	3.1e-3	7.2e-3
-0.3	2.6e-3	1.1e-2	9.3e-3	1.8e-2	2.8e-3	5.2e-3
0.1	9.1e-2	8.2e-2	5.6e-2	3.5e-2	9.4e-2	1.1e-1
0.5	1.9e-2	7.2e-2	1.4e-2	1.4e-2	4.7e-3	3.1e-3
0.9	9.4e-3	1.1e-2	2.3e-2	1.1e-1	3.3e-3	1.6e-3

Gemiddelde simulatietijd: 0.053 seconde
Gemiddelde fout: 0.028 rad

GCC-SCOT:

Tabel 5: Fout in rad tussen de berekende en werkelijke hoek voor SCOT

	-1.1	**-0.7**	**-0.3**	**0.1**	**0.5**	**0.9**
-1.1	7.3e-2	5,1e-2	3,3e-2	5.9e-2	8.2e-2	7.4e-2
-0.7	2.6e-2	7.3e-2	8.4e-2	1.6e-2	3.0e-2	6.0e-2
-0.3	2.7e-2	3.3e-2	4.2e-2	2.3e-2	3.4e-2	2.2e-2
0.1	1.0e-1	9.5e-2	8.9e-2	1.6e-1	1.1e-1	9.5e-2
0.5	1.2e-1	2.6e-2	1.5e-1	2.0e-1	3.3e-2	1.6e-1
0.9	2.8e-2	2.5e-2	8.9e-2	1.8e-1	4.8e-2	3.1e-2

Gemiddelde simulatietijd: 0.046 seconde
Gemiddelde fout: 0.071 rad

A.3.3 Conclusie

Uit deze simulaties is gebleken dat de weegfunctie SCOT 15% sneller is dan PHAT. De fout is echter ook anderhalf maal zo groot. Vandaar dat de keuze is gemaakt om door te gaan met de weegfunctie PHAT.

A.4 M-file voor scherpe signalen

Deze bijlage geeft uitleg over de geschreven code. Met onderstaande M-file is het mogelijk om uit gemeten data met de Sorama microfoonarray scherpe signalen te filteren en de richting ten opzichte van de microfoonarray te bepalen.

Inladen data-file

Het eerste deel van de M-file zorgt voor het opschonen van het geheugen, het inladen van de meetdata en het wegschrijven van informatie van de M-file naar de command window. Dit deel wordt alleen gebruikt om achteraf data te analyseren. Voor de *Soramadirectivitytool* wordt dit deel niet gebruikt.

```
clear all; clc; close all;
disp('————————————Thijs_4mics_top (18—07—2011)——————————Meting1_18062011————')
disp('      Vergelijken van eerste top. Hoek bepaald middels geometrie       ')
load('Meting1_15072011')
```

Coördineren meetdata

Het volgende deel zorgt voor het coördineren van de meetdata uit de data-file. De data dient eerst uit de celstructuur gehaald te worden voor bewerkingen. Aangezien er slechts 4 van de 64 microfoons worden gebruikt, dienen de microfoons aan de correcte meetdata verbonden te worden. Verder wordt de offset van het signaal afgehaald.

```
x1 = mappedwaveforms{1};              %Data uit celstructuur halen
x2 = mappedwaveforms{8};
x3 = mappedwaveforms{57};
x4 = mappedwaveforms{64};

avg_x1 = mean(x1);  x1 = x1—avg_x1;  %Offset weghalen
avg_x2 = mean(x2);  x2 = x2—avg_x2;
avg_x3 = mean(x3);  x3 = x3—avg_x3;
avg_x4 = mean(x4);  x4 = x4—avg_x4;
```

Vinden van pieken

Vervolgens wordt er in het signaal gezocht naar pieken. Indien de waarde van een sample hoger is dan *mph* (instelbare waarde), wordt deze herkend als een piek. Om reverberation (**zie hoofdstuk 12345678**) tegen te gaan, kan er vervolgens 2000 samples geen nieuwe piek gevonden worden. Het programma zoekt niet naar pieken die te dicht bij het begin of einde liggen. Voor deze pieken is er namelijk niet genoeg informatie om de verschiltijden te vinden.

```
k = 1;
indices(1,1) = —10000;
mph = 0.1;                            %minimale hoogte piek
for i = 101:length(x1)—100
    if (x1(i,1) > mph) && i > indices(k,1)+2000
        if isempty(find(x2(i—80:i+80,1)>mph, 1)) == 0 && ...
                isempty(find(x3(i—80:i+80,1)>mph, 1)) == 0 && ...
                    isempty(find(x4(i—80:i+80,1)>mph, 1)) == 0
            k = k+1;
            indices(k,1) = i;         %array voor opslaan pieken
        end
    end
end

if length(indices)<2          %Als er geen pieken worden gevonden in het signaal, wordt 'Geen piek gevonden
    disp(' ')                 %in meting' weggeschreven naar de command window. Het algoritme stopt.
    disp('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!')
    disp('!!!Geen piek gevonden in meting!!!')
    disp('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!')
```

Toewijzen ruimte in matrices

Indien er pieken zijn gevonden in het signaal, worden de (lege) matrices en vectoren aangemaakt om informatie in op te slaan. Als deze matrices nog niet zijn aangemaakt op het moment dat er data naar toe wordt geschreven, worden deze matrices pas op dat moment gevormd. Dit kost relatief veel tijd. Doordat de grootte van de matrices al bekend is op dit moment, kan deze tijd bespaard worden. Zoals te zien in regel 2-3 zijn de blokken 300 samples groot (100 samples vóór de piek en 200 samples na de piek worden gebruikt). **Vanaf dit moment worden nog slechts 2 van de 4 microfoons weergegeven in de code, aangezien deze er hetzelfde uitzien.**

```
else
    Pre_s = 100;        %Aantal samples v\'o\'or de piek
    Post_s = 200;       %Aantal samples na de piek

    x1block = zeros(k-1,Pre_s+Post_s+1);
    x2block = zeros(k-1,Pre_s+Post_s+1);

    avg_blockx1 = zeros(k-1,1);
    avg_blockx2 = zeros(k-1,1);

    in_top1 = zeros(k-1,1);  in_dal1 = zeros(k-1,1);
    in_top2 = zeros(k-1,1);  in_dal2 = zeros(k-1,1);
```

Informatie wegschrijven naar de blokken

De 300 samples rondom de piek worden in de lege matrices gezet. Vervolgens wordt per blok nog de offset weggehaald, om het vinden van de eerste top en dal (zie hoofdstuk 4) beter mogelijk te maken.

```
    for i = 1:k-1
        bl_pre = indices(i+1,1)-Pre_s;
        bl_post = indices(i+1,1)+Post_s;
        if bl_pre <= 0
            x1block(i,:) = [zeros(-bl_pre+1,1);x1(1:bl_post)];
            x2block(i,:) = [zeros(-bl_pre+1,1);x2(1:bl_post)];

        elseif bl_post > length(x1);
            x1block(i,:) = [x1(bl_pre:end);zeros(bl_post-length(x1),1)];
            x2block(i,:) = [x2(bl_pre:end);zeros(bl_post-length(x2),1)];

        else
            x1block(i,:) = x1(bl_pre:bl_post);
            x2block(i,:) = x2(bl_pre:bl_post);

        end

   end

    %% Offset per block weghalen
    for i = 1:k-1
        avg_blockx1(i,1) = mean(x1block(i,1:50));
        avg_blockx2(i,1) = mean(x2block(i,1:50));

        x1block(i,:) = x1block(i,:)-avg_blockx1(i,:);
        x2block(i,:) = x2block(i,:)-avg_blockx2(i,:);

    end
```

Vergelijken toppen en dalen

De blokken worden per microfoon met elkaar vergeleken. Elke microfoon wordt met elkaar vergeleken, zodat er voor 4 microfoons 6 microfoonparen overblijven. In onderstaande code staat slechts één microfoonpaar weergegeven, om het overzichtelijker te houden. De variabele *in_top12* bevat het sample-verschil tussen beide signalen.

```
%% Vergelijken toppen
for i = 1:k-1
    [~,in_top1(i,:)] = findpeaks(x1block(i,:),'minpeakheight',mph/4);
    [~,in_top2(i,:)] = findpeaks(x2block(i,:),'minpeakheight',mph/4);

end

%% Vergelijken dalen
for i = 1:k-1
    [~,in_dal1(i,:)] = findpeaks(-x1block(i,:),'minpeakheight',mph/4);
    [~,in_dal2(i,:)] = findpeaks(-x2block(i,:),'minpeakheight',mph/4);

end


in_top12 = in_top1-in_top2;
in_dal12 = in_dal1-in_dal2;
```

Verschiltijden bepalen

Indien er meer dan drie pieken in het signaal worden gevonden, worden de uitschieters van de verschiltijden verwijderd. In de praktijk kan namelijk voorkomen dat er één van de vijf verschiltijden sterk afwijkt. Met onderstaande code wordt deze uitschieter verwijderd, zodat deze niet meegenomen wordt voor de uiteindelijke TDOA.

```
if k-1 > 2      % Indien meer dan 3 pieken, twee uitschieters verwijderen voor beter resultaat.
    dT12temp = [in_top12;in_dal12];
    [~,max12] = max(dT12temp); dT12temp(max12) = [];
    [~,min12] = min(dT12temp); dT12temp(min12) = [];

else
    dT12temp = [in_top12;in_dal12];

end

%TDOA voor microfoonparen [in samples]
dT12 = mean(dT12temp);
dT13 = mean(dT13temp);
dT14 = mean(dT14temp);
dT23 = mean(dT23temp);
dT24 = mean(dT24temp);
dT34 = mean(dT34temp);
```

Bepalen hoek

Met behulp van onderstaande code wordt de hoek bepaald naar de geluidsbron. De code volgt de theorie die staat beschreven in hoofdstuk 2.3. Uiteindelijk wordt de hoek weergegeven in een polaire plot (zie figuur 16).

```
    Fs = 46875;          %Samplefrequentie
    c = 340.12;          %Snelheid geluid
    d = 0.14;            %Lengte zijde microfoonarray

    dT = [0 ;dT12 ;dT13 ;dT14];
    [~,first_mic] = max(dT);    %Bepalen bij welke microfoon het geluid het eerste is

    dT = [    0     dT12   dT13   dT14;
            -dT12    0     dT23   dT24;
            -dT13  -dT23    0     dT34;
            -dT14  -dT24  -dT34    0    ];

    if first_mic == 1
            Angs(1,1) = real( -asin((c*dT(2,1))/Fs/d) );
            Angs(1,2) = real( -acos((c*dT(4,1))/Fs/d) );

    elseif first_mic == 2
            Angs(1,1) = real( asin((c*dT(1,2))/Fs/d) );
            Angs(1,2) = real( acos((c*dT(3,2))/Fs/d) );

    elseif first_mic == 3
            Angs(1,1) = real( -asin((c*dT(4,3))/Fs/d)+pi );
            Angs(1,2) = real( -acos((c*dT(2,3))/Fs/d)+pi );

    elseif first_mic == 4
            Angs(1,1) = real( asin((c*dT(3,4))/Fs/d)-pi );
            Angs(1,2) = real( acos((c*dT(1,4))/Fs/d)-pi );

    end

    Ang = mean(Angs);   %Het middelen van de twee gevonden hoeken
    figure; polar([Ang Ang], [0 1],'r');

end
```

A.5 M-file voor zachte signalen

Een zacht signaal ziet er geheel anders uit dan een sterk signaal. Vandaar dat er een andere MATLAB-code gebruikt dient te worden. Het inladen van de data-file,het coördineren van de meetdata en het vinden van pieken in het signaal geschied op dezelfde wijze als bij sterke signalen (zie bijlage A.4).

A.5.1 Toewijzen ruimte in matrices

Voor de zachte signalen worden er blokken gevormd van 1000 samples. Deze waarde is zo gekozen dat het genoeg informatie over het signaal bevat (ook bij lage frequenties), maar dat het niet al te lang duurt om te verwerken.

```
x1 = x1(indices-500:indices+499);
x2 = x2(indices-500:indices+499);
x3 = x3(indices-500:indices+499);
x4 = x4(indices-500:indices+499);
```

A.5.2 Crosscorrelatie berekenen

In de *Signal Processing Toolbox* van MATLAB zit een ingebouwde functie voor het berekenen van de crosscorrelatie tussen twee signalen. Deze functie 'xcorr' berekent de crosscorrelatie aan de hand van onderstaande formule (14). Hier zijn x en y de twee verschillende signalen van het microfoonpaar.

$$R_{xy}(m) = E(x1_{n+m}y_n^*) = E(x_n y_{n-m}^*) \tag{14}$$

In de gevonden faseverschuivingen wordt vervolgens gezocht naar het maximum (aangezien deze de faseverschuiving weergeeft) en in de variabele dT gestopt. Voor dit algoritme worden ook de 4 microfoons gebruikt op de hoeken van de microfoonarray. Om het overzichtelijk te houden is er slechts één van de zes microfoonparen weergegeven in de code.

```
%Berekenen crosscorrelatie ('coeff' normaliseert de crosscorrelatie)
R12total = xcorr(x1,x2,'coeff');

%% Afkappen van niet interessante verschiltijden
R12 = R12total(length(x1)-Dif_sample:length(x1)+Dif_sample,:);

%% Vinden van maximum in crosscorrelatie
[~,mc12] = max(R12);
dT12 = (mc12-(Dif_sample+1))';
```

Bepalen hoek

Met behulp van de TDOA wordt de hoek bepaald op dezelfde manier als voor een zacht signaal. Uiteindelijk wordt de gevonden hoek geplot in een polaire plot.

A.6 Experiment1

Tijdens het ontwikkelen van het algoritme voor scherpe tonen zijn er meerdere metingen gedaan. In deze bijlage wordt dit experiment omschreven en wordt er een conclusie getrokken uit de resultaten.

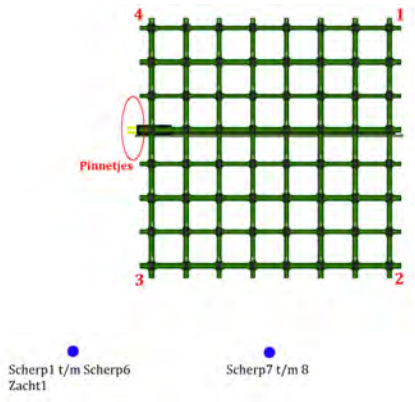
A.6.1 Opstelling

Tijdens de experimenten is de microfoonarray horizontaal vlak op de tafel gelegd in de kantoorruimte van Sorama. Deze ruimte is redelijk rumoerig door de computers die aan staan, het tikken op toetsenborden en het praten van de mensen in de omgeving. De microfoonarray lag met de pinnetjes links (zie figuur 16, om overeen te komen met de polaire plot van MATLAB. De microfoonarray is verbonden via drie ethernet kabels met het circuit board, dat weer verbonden is met een voeding.

De gebruikte microfoonarray is de '64 Channel DMADAQ Prototype' van Sorama die weergegeven is in figuur 23.

A.6.2 Metingen

Tijdens het experimenteren zijn er meerdere files opgeslagen van metingen uit het verleden. De metingen zullen in dit deel beschreven worden. In figuur 31 staan de metingen weergegeven. De blauwe punten geven de locatie weer van de geluidsbron.



Figuur 31: Weergave van de meetpunten voor de experimenten

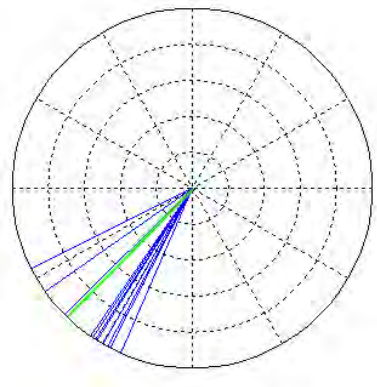
Zoals te zien is in bovenstaand figuur, zijn er metingen op de 1-3 (microfoons) lijn en de verticale lijn. In totaal zijn er zeven metingen verricht.

Resultaten

Voor elke meting is elke hoek geanalyseerd die uit elke piek wordt berekend met het algoritme. De resultaten van de vier metingen (Scherp1 t/m Scherp4) voor elke piek staan weergegeven in tabel 6. De ingevoerde (juiste) hoek is -2.36 rad (oftewel -135 graden), aangezien al deze metingen op de 1-3 lijn liggen. Aangezien niet bij elke meting hetzelfde geluid is gemaakt, heeft niet elke meting hetzelfde aantal pieken. In figuur 32 staan deze hoeken weergegeven in een polaire plot. De groene lijn is de ideale lijn op -2.36 rad.

Tabel 6: Berekende hoeken uit metingen scherpe tonen voor -2.36 rad

Meting:	**Piek nr.**	**Hoek [rad]**	**Fout [rad]**
Scherp1	1	-2.14	0.22
	2	-2.36	0.00
	3	-2.17	0.19
	4	-2.06	0.30
	5	-2.17	0.19
	6	-2.14	0.22
	7	-2.36	0.00
Scherp2	1	-2.16	0.20
	2	-2.13	0.23
	3	-2.09	0.27
Scherp3	1	-2.05	0.31
	2	-1.98	0.38
	3	-2.36	0.00
Scherp4	1	-2.68	0.32
	2	-2.68	0.32
	3	-2.53	0.17

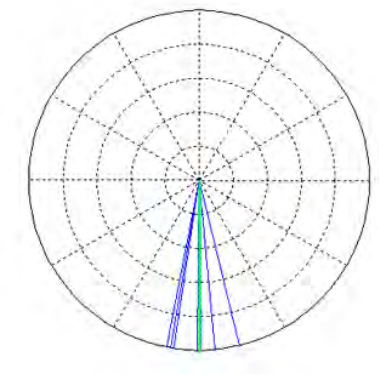


Figuur 32: Polaire plot van berekende hoeken voor scherp signaal

Behalve de metingen op de 1-3 lijn, zijn er voor scherpe tonen ook nog 3 metingen op de verticale lijn. De resultaten van deze metingen staan weergegeven in tabel 7. Het ideale resultaat zou hier -1.57 rad zijn (-90 graden). Ook staat de polaire grafiek van deze metingen gegeven in figuur 33.

Tabel 7: Berekende hoeken uit metingen scherpe tonen voor -1.57 rad

Meting:	**Piek nr.**	**Hoek [rad]**	**Fout [rad]**
Scherp5	1	-1.33	0.24
	2	-1.48	0.09
Scherp6	1	-1.56	0.01
Scherp7	1	-1.58	0.01
	2	-1.74	0.17
	2	-1.72	0.15
	3	-1.76	0.19



Figuur 33: Polaire plot van berekende hoeken voor scherp signaal

Conclusie

Uit bovenstaande resultaten blijkt dat het algoritme de richting redelijk goed weet te vinden. Tijdens de metingen was het echter niet duidelijk of het geluid ook precies op de lijnen werd gemaakt. Het is dan ook moeilijk om een conclusie te trekken uit deze resultaten.