# Hoe precies kan er realtime een Android-apparaat akoestisch gelokaliseerd worden?

Jos Bonsink & Mustafa Karaalioglu

7 juli 2013

## 1 Inleiding

Aan het begin van het Net-Centric Computing project hadden enkele groepen besloten met robots te gaan werken. Hierbij was het nodig om realtime de lokale posities van de robots te kunnen bepalen. Tegenwoordig wordt positiebepaling voornamelijk gedaan met behulp van GPS. Echter, is dit niet overal beschikbaar. Tevens is GPS slechts nauwkeurig tot op enkele meters. GPS is dus niet geschikt voor positiebepaling in kleine ruimtes. Andere groepen kozen ervoor om gebruik te maken van *image vision* waar de robots altijd in beeld moet zijn. Dit artikel onderzoekt de mogelijkheden om realtime [7] posities akoestisch te bepalen.

Er is reeds onderzoek gedaan naar de applicatie van geluid om positie te bepalen. Enzo Mumolo, Massimiliano Nolich en Gianni Vercelli hadden een algoritme ontwikkeld voor realtime lokalisatie voor het gebruik van een microfoonarray in luidruchtige omstandigheden. Zij haalden een precisie van ongeveer 20 cm voor 90% van alle gevallen in luidruchtige omstandigheden [3]. Een microfoonarray plaatsen op een robot is duur en log. Er is ook onderzoek geweest naar goedkopere oplossingen zoals een telefoon.

Microsoft Asia had onderzoek gepleegd naar het gebruik van telefoons om afstanden te bepalen. Ze hadden een algoritme ontwikkeld dat tot een paar centimeter precies was in stille omgevingen [4]. Vervolgonderzoek breidde dit uit tot volledige positiebepaling met behulp van geluid en andere sensoren zoals een gyroscoop. Zij behaalden een nauwkeurigheid van 13.9 cm voor 90% van de gevallen [5].

Zou dit preciezer kunnen? Hoe precies kan er realtime een Android-apparaat akoestisch gelokaliseerd worden? De verwachting is dat een nauwkeurigheid van 10 cm te behalen moet zijn. Dit is gebaseerd op de volgende feiten: de samplerate van een microfoon is 44100 Hz, de geluidssnelheid is 340 meter per seconde. Het vermoeden is dat een piepgeluid binnen 25 samples herkend kan worden. Dit vertaalt zich naar een precisie van 10 cm.

Er zal een Android-applicatie worden ontwikkeld waarin realtime piepgeluiden gegenereerd en gedetecteerd kunnen worden binnen 25 samples. Het artikel is als volgt ingedeeld: paragraaf 3 beschrijft de gebruikte materialen en ontwikkelde methoden om realtime accu-

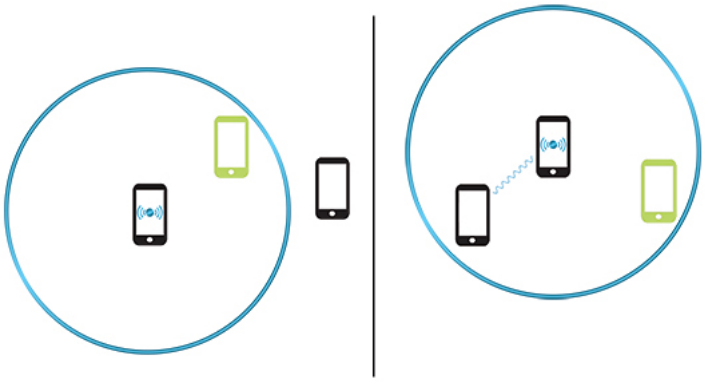
rate akoestische lokalisatie uit te voeren, in paragraaf 4 worden de resultaten getoond en tot slot worden in paragraaf 5 de resultaten geëvalueerd en enkele conclusies getrokken.

# 2 Materialen en methoden

De experimenten met de Android-applicatie werden uitgevoerd op HTC Desire C telefoons met Android 4.0.2. Om positiebepaling mogelijk te maken met deze telefoons moesten er drie problemen worden opgelost: Bluetooth communicatie, audio detectie en afstandsmeting.

## 2.1 Bluetooth

Er zijn minstens drie Android-apparaten nodig om lokaal de positie van één Android-apparaat te kunnen bepalen. De benodigde communicatie verliep via een peer-to-peer [6] netwerk van Bluetooth [2] verbindingen.

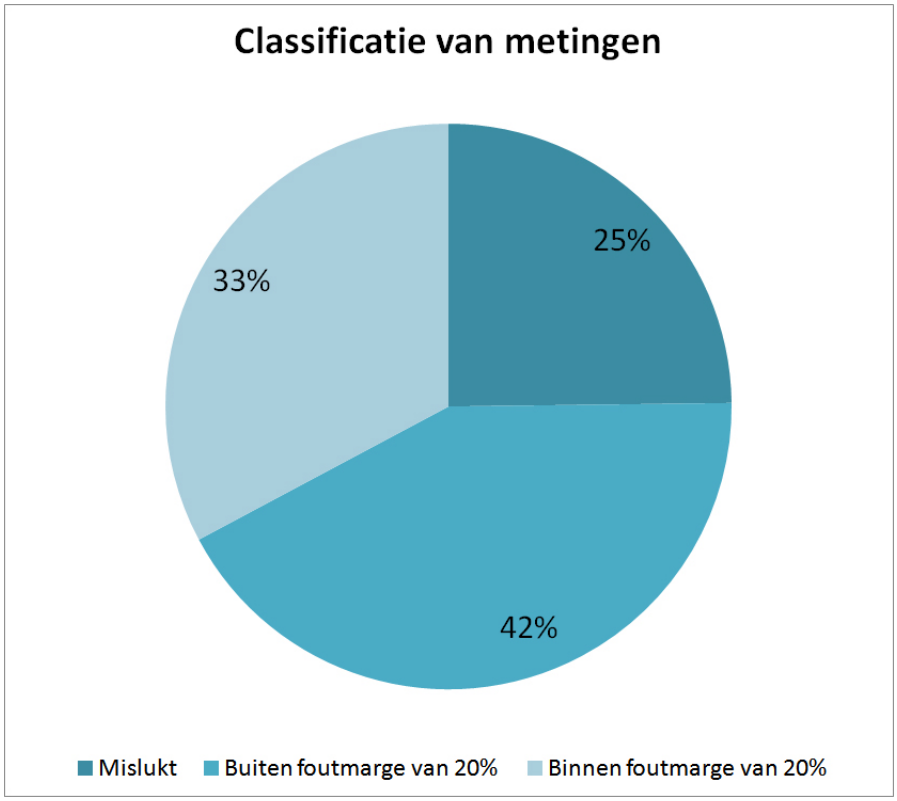


Figuur 1: opzetten van peer-to-peer Bluetooth-netwerk

Alle deelnemende apparaten moesten op *Bluetooth Discoverable* worden gezet voor een oneindige periode. Het proces startte bij één node die zocht naar naburige Bluetooth apparaten, geïllustreerd in figuur 1, met het Bluetooth Discovery proces. Dit proces werd telkens gestopt wanneer er één apparaat werd gevonden. Vervolgens werd er geprobeerd met dit apparaat te verbinden, in figuur 1 geïlustreerd als de groene telefoon. Als dit mislukte, werd Bluetooth Discovery opnieuw opgestart.

Indien het verbinden van beide apparaten was geslaagd, werd Bluetooth Discoverable uitgezet. De laatst toegevoegde node ging zoeken naar nieuwe toevoegingen voor het netwerk. Dit proces herhaalde zich totdat er na een aantal Bluetooth Discoveries geen nieuwe apparaten waren gevonden. Hierna was het dus niet meer mogelijk om het netwerk uit te breiden. Alle nodes in het netwerk waren niet meer Discoverable.

Elke nieuwe node in het netwerk ontving van zijn *parent* een lijst met de reeds bekende mac-adressen. De node voegde zijn eigen mac-adres aan deze lijst toe. Vervolgens stuurde



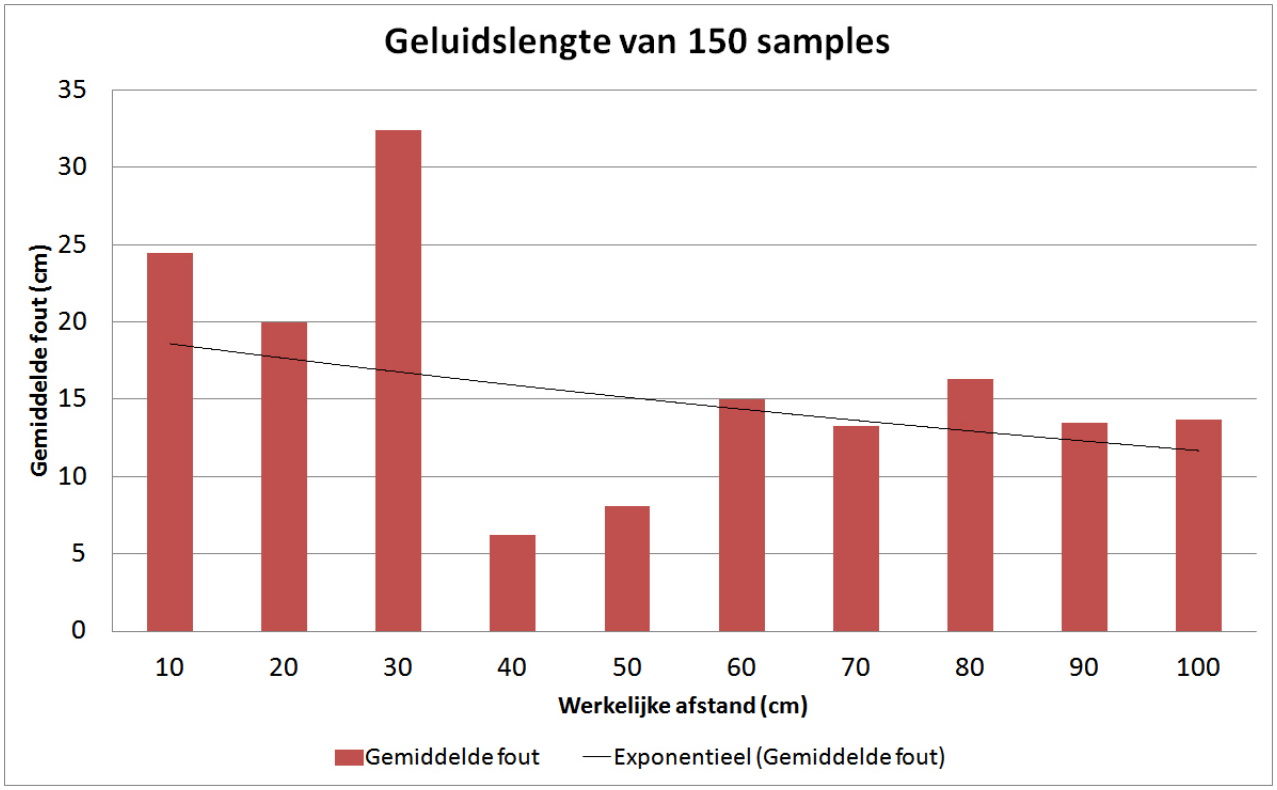

Figuur 2: classificatie van meetpogingen.

hij zijn eigen mac-adres naar zijn parent, die dit adres aan zijn eigen lijst toevoegde. Het mac-adres propageerde door het netwerk totdat het mac-adres bij de root node was terechtgekomen. Een lijst van alle mac-adressen was gelijk en beschikbaar op elk apparaat.

## 2.2 Audio detectie

Voor het realtime detecteren van piepgeluiden moest er constant geluid opgenomen én geanalyseerd worden. De audiohardware bufferde continu een halve seconde aan audio. In een aparte thread werden er steeds twintig samples uit deze buffer opgehaald. Op deze samples werd een Fast Fourier Transformatie(FFT) [1] toegepast. Hiermee werd het geluidssignaal als functie van tijd omgezet naar een complexe functie van amplitude en fase als functie van frequentie.

De gegenereerde piepgeluiden bestonden uit een sinusoïde met een frequentie van 8820 Hz. Na de FFT werd de lengte van het complexe getal corresponderend met de frequentie van het piepgeluid vergeleken met een drempelwaarde. Wanneer de lengte de drempelwaarde overschreed, werd het geluid herkend als een piep.

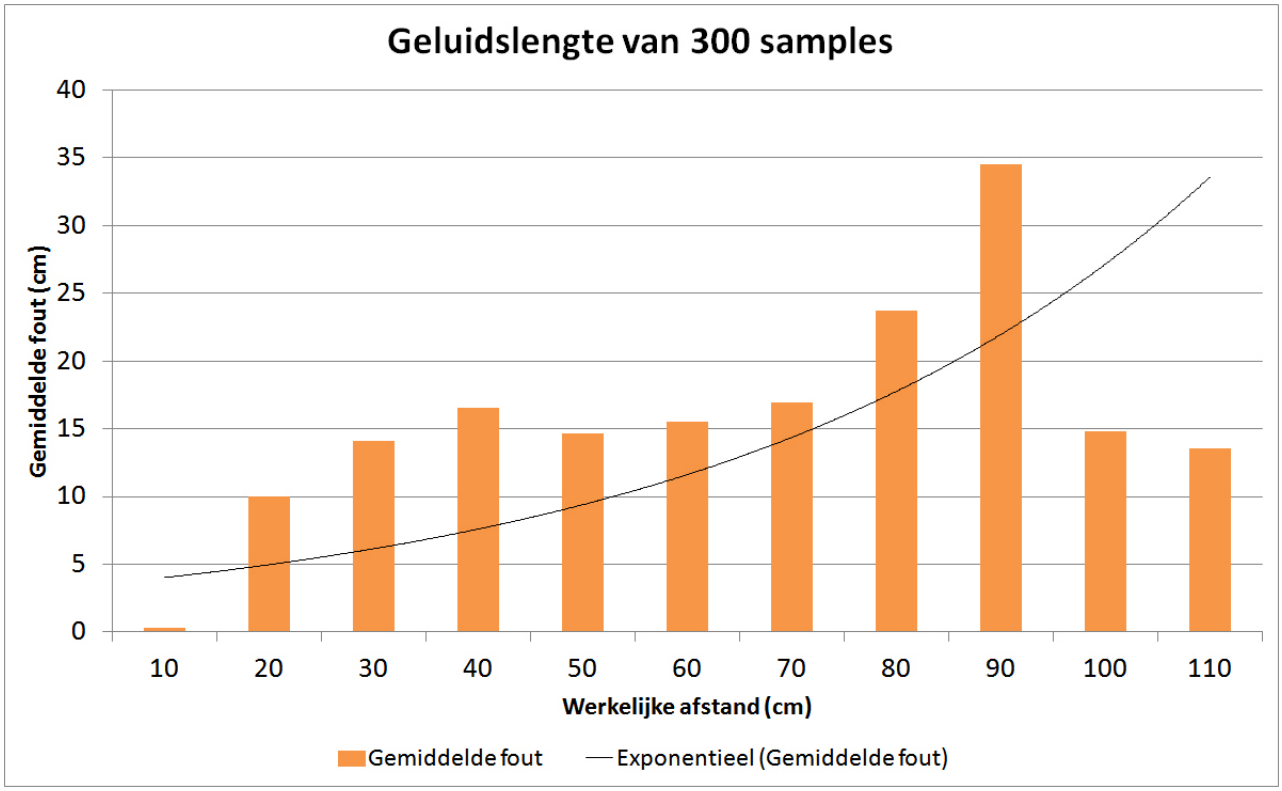



Figuur 3: gemiddelde fout bij piepgeluid met geluidslengte van 150 samples.

## 2.3 Afstandsmeting

De afstand tussen twee Android-apparaten werd bepaald door het tijdsverschil tussen het verzenden en ontvangen van gegenereerde piepgeluiden te meten. Uit de eerder genoemde lijst van mac-adressen in het Bluetooth-netwerk werden een *master* en *slave* uitgekozen aan de hand van de volgorde van de lijst. De master verzocht via Bluetooth de slave te antwoorden op zijn piepgeluid. Na bevestiging van de slave, speelde de master een piepgeluid af.

Het tijdsverschil tussen het versturen en ontvangen van piepgeluiden kon worden gemeten door het aantal samples tussen twee piepgeluiden te gebruiken. Het twee keer achterelkaar detecteren van hetzelfde piepgeluid, werd voorkomen door een korte periode na een detectie niks te detecteren. De looptijd kon exact worden verkregen door te corrigeren voor de verwerkingstijd die nodig is bij het afspelen van geluid. Voor de master werd het aantal samples tussen de detectie van zijn eigen piep en dat van de respons gemeten. Bij de slave werd het verschil gemeten tussen de detectie van het eerste piepgeluid en dat van zichzelf. Dit verschil werd via Bluetooth naar de master verstuurd. De master trok deze waarde af van de gemeten totaal aantal samples tussen de twee piepgeluiden. Aan de hand van de samplerate (44100Hz) kon de tijdsduur worden bepaald, hiermee kon vervolgens de afstand worden berekend.




Figuur 4: gemiddelde fout bij piepgeluid met geluidslengte van 150 samples.

# 3 Resultaten

Figuur 3 en 4 tonen de gemiddelde fouten bij de afstandsmetingen voor geluidslengtes van respectievelijk 150 en 300 samples.

Alle geregistreerde afstandsmetingen waren in drie verschillende klassen ingedeeld. In figuur 2 wordt weergegeven dat 25% van alle metingen niet konden worden voltooid. Deze pogingen werden geclassificeerd als mislukt. Geslaagde metingen werden geclassificeerd op basis van precisie: 33% van de metingen zaten binnen een foutmarge van 20% en 42% erbuiten.

Het aantal metingen dat correct is binnen een foutmarge van 10 cm is 21.7% en 23.3% voor piepgeluiden met een lengte van respectievelijk 150 en 300 samples.

# 4 Discussie

Het meetproces is te fragiel gebleken en is niet geschikt voor nauwkeurige positie bepaling. Uit figuur blijkt dat 25% van de afstandsmetingen mislukken, dit kan worden veroorzaakt doordat een piepgeluid ergens in het proces niet wordt herkend. Het missen van een piepgeluid kan wel worden herkend en het proces kan dan opnieuw worden opgestart.

Uit de resultaten blijkt ook dat de geluidslengte invloed heeft op de prestaties van het

systeem. Uit figuren 3 en 4 blijkt dat geluiden van een langere lengte op kortere afstanden beter presteren dan geluiden met een kortere lengte. De oorzaak hiervan kan nog niet worden gevonden. Er zijn meer experimenten hiervoor nodig, dit zou eventueel in een vervolgonderzoek kunnen worden uitgevoerd.

In voorgaand onderzoek [5] waren in 90% van de gevallen afstanden tot 13.9 cm nauwkeurig te meten. Onze hypothese was dat we een nauwkeurigheid van 10 cm zouden kunnen halen. Dit was alleen mogelijk in 16.7% van de gevallen. De gebruikte materialen in dit onderzoek waren niet toereikend om tot dezelfde of betere resultaten gekomen. Een beter resultaat zou kunnen worden behaald door telefoons te gebruiken die over meerdere microfoons beschikken.

Realtime akoestische positiebepaling is zeker mogelijk met het juiste materiaal. De HTC Desire C toestellen lijken hier niet geschikt voor te zijn.

# 5 Nawoord

Mustafa had zich tijdens het project gericht op het audio gedeelte en Jos op het Bluetooth gedeelte.

# Referenties


[1] Ronald Newbold Bracewell and RN Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[2] Jaap C Haartsen. The bluetooth radio system. *Personal Communications, IEEE*, 7(1):28–36, 2000.

[3] Enzo Mumolo, Massimiliano Nolich, and Gianni Vercelli. Algorithms for acoustic localization based on microphone array in service robotics. *Robotics and Autonomous systems*, 42(2):69–88, 2003.

[4] Chunyi Peng, Guobin Shen, Yongguang Zhang, Yanlin Li, and Kun Tan. Beepbeep: a high accuracy acoustic ranging system using cots mobile devices. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 1–14. ACM, 2007.

[5] Jian Qiu, David Chu, Xiangying Meng, and Thomas Moscibroda. On the feasibility of real-time phone-to-phone 3d localization. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 190–203. ACM, 2011.

[6] Rüdiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 101–102. IEEE, 2001.

[7] John A. Stankovic. Misconceptions about real-time computing: A serious problem for next-generation systems. *Computer*, 21(10):10–19, 1988.