Masterarbeit

Sound of Interest
Ein Ballspielroboter hört stereo

Michel Bartsch
Matrikel-Nr.: 2363665
Studiengang: Informatik

Erster Gutachter: Udo Frese
Zweiter Gutachter: Christian Mandel
Betreuer: Dennis Schüthe

2. Januar 2015

Inhaltsverzeichnis

Abstract

Diese Arbeit beschreibt ein System zum Stereohören, mit dem ein Ballspielroboter ausgestattet wird (siehe Abbildung 1). Damit reagiert er auf interessante Geräusche, indem er sich zu ihnen hin dreht. Neben den verbreiteten Algorithmen zur Lokalisierung von Signalquellen wurden im Rahmen dieser Arbeit auch verschiedene Methoden für höhere Robustheit und Präzision gegenüber Störgeräuschen implementiert und unter realistischen Bedingungen getestet. Darunter ein Phase Only Matched Filter[1] sowie ein neues adaptives Filter, welches die Häufigkeit von Frequenzen lernt und das Signal im Frequenzraum manipuliert, um dort oft auftretende Frequenzen abzuschwächen. Letzteres stellte sich als geeigneter für den praktischen Einsatz in diesem Gebiet heraus.

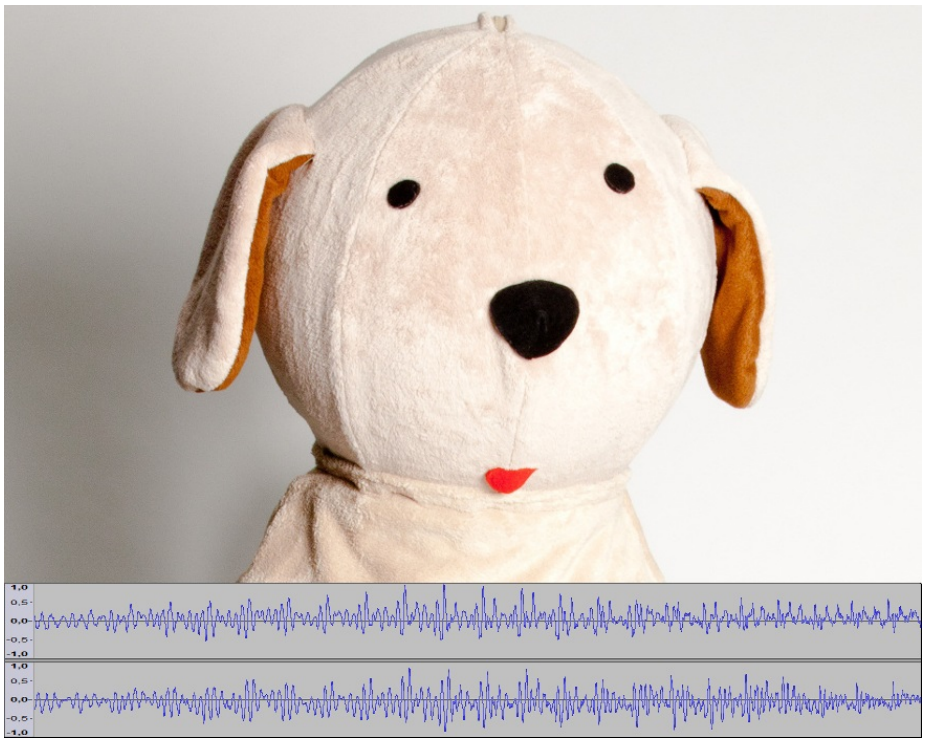

Abbildung 1: Kopf des Roboters Doggy, darunter ein Ausschnitt aus dem Signal vom linken und rechten Mikrofon.

Kapitel 1

Einleitung

In diesem Kapitel wird der Ballspielroboter näher beschrieben. Hierzu zählen das Stereohören, die damit einhergehenden Schwierigkeiten und die angestrebten Ziele.

1.1 Doggy der Roboter

Der in dieser Arbeit verwendete Ballspielroboter heißt Doggy (siehe Abbildung 1.1). Er ist ein Prototyp der AG Multisensorische Interaktive Systeme des Fachbereichs 3 (Mathematik und Informatik) an der Universität Bremen. Er hatte bereits einen sehr ähnlichen Vorgänger mit dem Namen Piggy. Ein Nachfolger wird bereits gebaut, steht zum Zeitpunkt dieser Arbeit aber noch nicht zur Verfügung.

Doggy soll ein Roboter mit Unterhaltungswert sein, inspiriert von dem Kinderspiel „Schweinchen in der Mitte", welches auch den Namen seines Vorgängers erklärt. Dabei werfen sich zwei oder mehr Personen einen Ball gegenseitig zu, während zwischen ihnen eine Person steht, die versucht den Ball zu fangen. Dieses Schweinchen in der Mitte soll der Roboter sein. Doggy ist zwar nicht in der lage den Ball zu fangen, doch er soll ihn zum Werfer zurück schlagen können.

Damit er das kann, muss er den Ball erst einmal erkennen können. Dazu ist Doggy mit zwei Kameras ausgestattet, mit denen er stereo sehen kann. Damit beobachtet er den Ball, bis er eine Schätzung über dessen zukünftige Flugbahn machen kann. Dann bewegt er seinen gesamten Oberkörper durch zwei Freiheitsgrade in die Flugbahn des Balles, sodass er ihn mit seinem Kopf möglichst zum Werfer zurück schlägt[2].

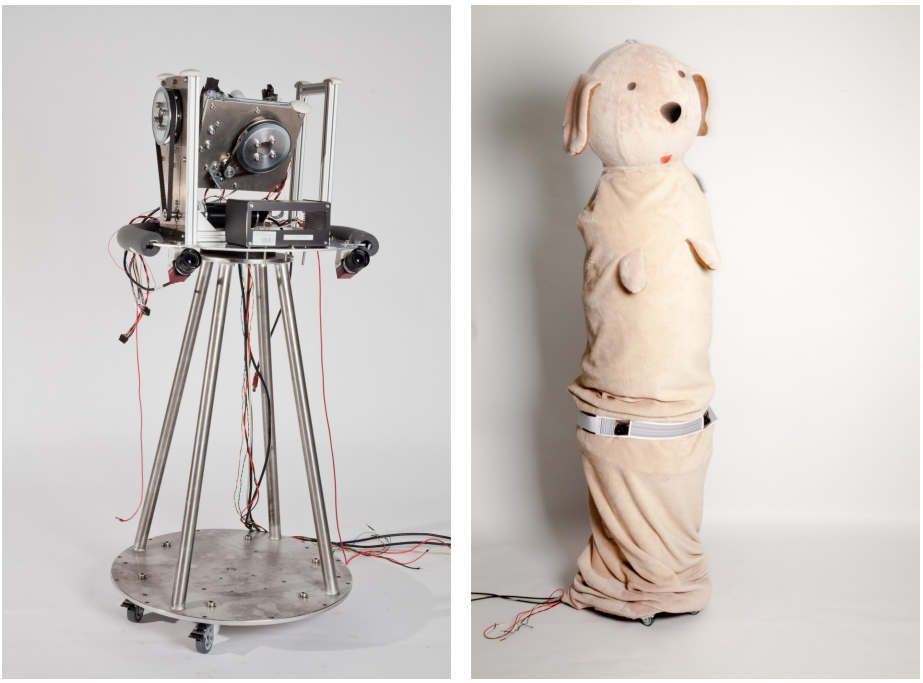

Abbildung 1.1: Doggys technisches Innenleben links, rechts mit aufgesetztem Oberteil und Kostüm.

1.2 Zielsetzung

Ziel dieser Arbeit ist es, den Ballspielroboter Doggy mit zwei Mikrofonen auszustatten sowie mit der nötigen Software, um interessante Geräusche als solche zu erkennen, die Richtung aus der sie kommen zu berechnen und den Roboter in diese Richtung zu drehen. Diese Erweiterungen des Roboters zum Stereohören sollen in seinen typischen Einsatzsituationen funktionieren. Dazu gehört zum Beispiel der Einsatz auf Festen oder mit einer großen Zahl von lauten Kindern. In diesen Fällen soll Doggy sich der Situation anpassen, um sich nicht permanent in alle Richtungen zu drehen, aus denen er etwas hört. Zudem muss die Lokalisierung von Geräuschen robust gegen Störgeräusche sein.

Wann ein Geräusch interessant ist, wurde im Vorfeld dieser Arbeit nicht genau definiert. Es gibt aber Beispiele, wie ein Klatschen auf einem Fest, eine rufende Person oder ein Räuspern in einem leisen Saal. Ziel ist es, dass der Roboter auf die selben Geräusche aufmerksam wird wie ein Mensch.

Störgeräusche, gegen welche die Lokalisierung robust sein soll, können vier Ursachen haben. Zum einen gibt es die allgemeine Geräuschkulisse an Doggys typischen Einsatzorten, also viele weniger auffällige Geräusche, auf die er nicht reagieren soll. Des Weiteren kann ein lautes Geräusch von einer nahen Wand widerhallen und so ein Echo entstehen, das nicht aus der ursprünglichen Richtung kommt. Hinzu kommen Störgeräusche, die der Roboter selbst durch seine Motoren erzeugt. Diese können sich im schlimmsten Fall als Vibrationen durch das Gestänge des Roboters direkt auf die Mikrofone übertragen. Zuletzt bleibt noch das normale Rauschen der Mikrofone, welches sie auch in einem vollkommen

stillen Raum aufzeichnen würden.

Kapitel 2

Konzepte

In diesem Kapitel werden die Ideen und Konzepte dieser Arbeit mit ihrem mathematischen Hintergrund beschrieben. Zunächst werden Kriterien für interessante Geräusche erläutert, anschließend wird darauf eingegangen, wie durch den zeitlichen Versatz zweier Signalen die Richtung einer Geräuschquelle bestimmt werden kann. Ebenfalls werden Methoden vorgestellt, um diesen zeitlichen Versatz zu ermitteln, sowie Filter, um das Ergebnis zu verbessern.

2.1 Kriterien für interessante Geräusche

Interessante Geräusche sind für diese Arbeit nicht genau definiert worden, es sollen aber Geräusche sein, auf die auch ein Mensch reagieren würde. Dabei hängt dies stark von der Umgebung ab. In einem stillen Saal wäre bereits ein lautes Räuspern interessant, während auf einem lauten Fest schon jemand kräftig klatschen oder rufen müsste.

Ein offensichtliches Merkmal von interessanten Geräuschen ist ihre Lautstärke. Damit Geräusche interessant sein können, müssen sie in dieser Arbeit einen Schwellwert einer bestimmten Lautstärke übersteigen. Dieser Schwellwert wird im Folgenden als Noise-Level bezeichnet: Alle Geräusche unterhalb dieser Lautstärke werden als Störgeräusche betrachtet. Der Noise-Level muss sich der Lautstärke in der Umgebung des Roboters anpassen. In einer lauten Umgebung muss er also höher liegen als in einer leisen. Die Lösung bieten Formeln, mit der sich der Noise-Level über die Zeit der Lautstärke anpasst.

$$v_0 = \max\{l_{\max}, r_{\max}\}\ F \tag{2.1}$$

$$v_m = v_{m-1}\ (1-\alpha) + \max\{l_{\max}, r_{\max}\}\ F\ \alpha \tag{2.2}$$

Der Noise-Level v_0 aus der ersten Messung ergibt sich aus den Variablen $l_{\max}$ und $r_{\max}$, der maximalen Lautstärke gemessen zum aktuellen Zeitpunkt vom linken und rechten Mikrofon und dem konstanten Faktor F, der typischerweise zwischen 1.0 und 2.0 liegt, um den Noise-Level höher zu setzen(2.1).

Mit jeder neuen Messung wird durch ein gewichtetes Mittelwert-Filter[3](2.2) ein neuer Noise-Level v_{m} aus seinem Vorgänger $v_{\mathrm{m\text{-}1}}$ und den aktuellen Messwerten berechnet. Der Faktor α ist eine Konstante und bestimmt, wie schnell sich

der Schwellwert anpasst. Bei $\alpha = 1$ würde sich der Noise-Level immer nur aus der aktuellen Messung ergeben, mit $\alpha = \frac{1}{2}$ läge der neue Noise-Level in der Mitte zwischen dem vorherigen und dem aktuellen. Je mehr sich α Null annähert, desto langsamer passt sich der Noise-Level neuen Messungen an, also beispielsweise einer lauter werdenden Umgebung. Ist α zu hoch, kann der Noise-Level durch kurzzeitige Änderungen der Geräuschkulisse zu stark beeinflusst werden. Um sich für einen sinnvollen Wert für α zu entscheiden, muss die Anzahl der Messungen pro Sekunde betrachtet werden und über welchen Zeitraum der Wert sich anpassen soll. Solange $\alpha < 1$ ist, werden einmal eingeflossene Messwerte den Noise-Level auch in allen Folgeschritten beeinflussen, jedoch läuft ihr Einfluss auf das Ergebnis mit der Zeit gegen Null.

Der Roboter Doggy soll sich durch interessante Geräusche von der Seite nicht zu sehr von einem Werfer ablenken lassen. Daher sollen Geräusche nur dann interessant genug sein, dass Doggy sich zu ihnen hin dreht, wenn diese innerhalb eines gewissen Öffnungswinkels vor ihm liegen. Aus Gründen, die bei der Lokalisierung durch zeitlichen Versatz (Kapitel 2.2) erläutert werden, liegt der maximale Öffnungswinkel bei 180°. Um aber Geräusche von der Seite zu ignorieren, sollte ein interessantes Geräusch in einem kleineren Öffnungswinkel liegen.

Geräusche unterscheiden sich neben ihrer Lautstärke vor allem durch die in ihnen enthaltenen Frequenzen. Betrachtet man beispielsweise einzelne Töne, so besteht der Unterschied zwischen einem hohen und einem tiefen Ton darin, dass der hohe Ton eine höhere Frequenz hat. Bei einer höheren Frequenz haben die Schallwellen in der Luft eine kürzere Wellenlänge, ihre Ausbreitungsgeschwindigkeit bleibt die Schallgeschwindigkeit. Für die meisten natürlichen Geräusche und besonders für die menschliche Sprache gilt, dass sie aus vielen verschiedenen Frequenzen zusammengesetzt sind.

Wenn sich zwei Geräusche durch die in ihnen enthaltenen Frequenzen unterscheiden lassen und das eine Geräusch ständig zu hören ist, während das andere nur selten auftaucht, dann sollte das seltene Geräusch als solches erkannt werden und ein stärkeres Interesse erzeugen. Dies ist besonders sinnvoll, wenn das häufige Geräusch von den Motoren des Roboters selbst stammt und somit sehr häufig und uninteressant sein kann. Dieser Aspekt zum Interesse an einem Geräusch wird in dieser Arbeit von dem Frequenzen lernenden Filter (Kapitel 2.5.4) aufgegriffen.

2.2 Lokalisierung durch zeitlichen Versatz

Angenommen es gibt zwei Mikrofone mit bekannter Position zueinander und ein Geräusch wird von einer Geräuschquelle erzeugt, deren Position unbekannt ist. Wird das Geräusch von beiden Mikrofonen aufgezeichnet, erlaubt dies Rückschlüsse auf die Richtung, aus der das Geräusch kam. Auf einer Ebene, die durch beide Mikrofone und die Geräuschquelle verläuft, lässt sich dann die mögliche Position der Geräuschquelle auf einen unbekannten Punkt auf einer Ortskurve begrenzen[4] (siehe Abbildung 2.1).

Dies ist möglich, wenn man den zeitlichen Versatz kennt, mit dem das Geräusch die beiden Mikrofone erreicht hat. Wenn sich das Geräusch mit gleichbleibender Geschwindigkeit ausgebreitet hat, lässt sich aus dem zeitlichen Versatz ein Entfernungsunterschied der Geräuschquelle zu den beiden Mikrofonen

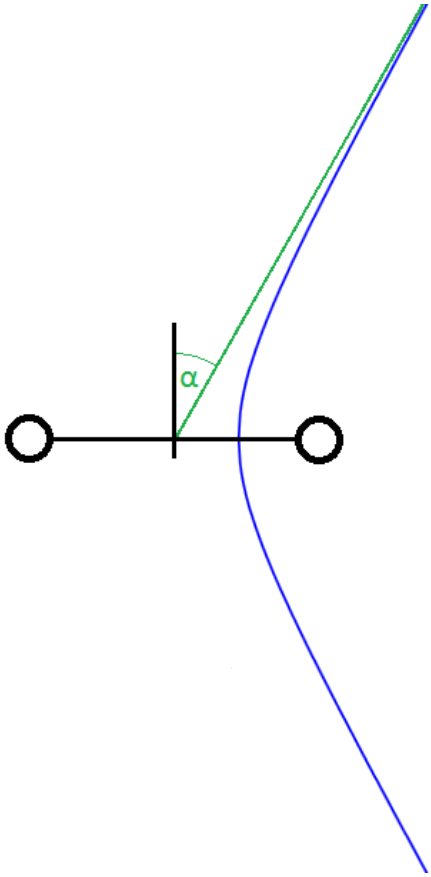

Abbildung 2.1: Kreise: Mikrofone am Roboter, blau: Ortskurve der Geräuschquelle, grün: Gerade zur Quelle in unendlicher Entfernung.

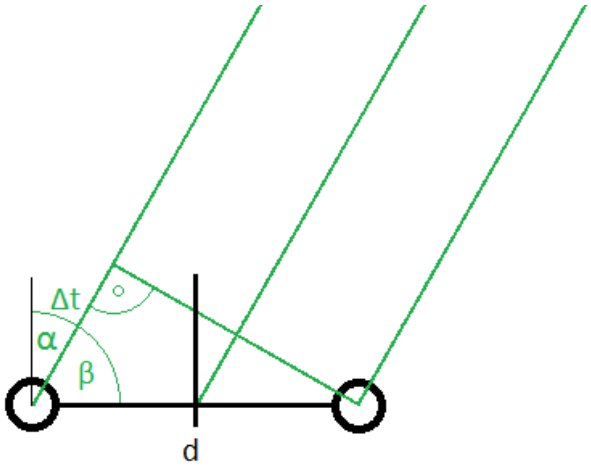

Abbildung 2.2: Situation und Geräusch wie in Fig. 2.1, Geräusch kommt angenommen aus unendlicher Entfernung.

berechnen. Dabei handelt es sich um den Entfernungsunterschied und nicht um die Entfernung zwischen Mikrofonen und Geräuschquelle. Zeichnet man nun alle Punkte auf, für die derselbe Entfernungsunterschied zu den beiden Mikrofonen besteht, erhält man die besagte Ortskurve, auf der die Geräuschquelle liegt.

Je weiter sich die Ortskurve von den Mikrofonen entfernt, desto mehr nähert sie sich einer von zwei Halbgeraden an, welche beide an der Mitte zwischen den Mikrofonen ihren Endpunkt haben. Sind die Mikrofone links und rechts an einem Roboter befestigt, dann ragt eine der Geraden in die vorderen 180° und die andere in die 180° hinter dem Roboter. Es ist also mit zwei Mikrofonen und nur dem zeitlichen Versatz eines Geräusches zwischen ihnen nicht möglich zu berechnen, ob das Geräusch vor oder hinter dem Roboter entstanden ist. Da Doggy nur Geräusche vor sich interessant finden soll, wird im Folgenden nur die nach vorne gerichtete Halbgerade betrachtet, was zu Fehlern bei Geräuschen hinter dem Roboter führt.

Bei einer Entfernung der Geräuschquelle zum Roboter, die etwa dem Abstand der beiden Mikrofone zueinander entspricht, ist die Annäherung der Ortskurve an die Halbgerade bereits stark genug, dass für den praktischen Einsatz auch nur noch die Halbgerade und nicht mehr die Ortskurve betrachtet werden kann. Dies ist dasselbe, als würde man annehmen, das Geräusch käme aus unendlicher Entfernung, da sich dort Ortskurve und Halbgerade treffen. Diese Annahme ist berechtigt, da ein Mensch beim Spielen mit dem Roboter mehrere Meter von diesem entfernt ist, also ein vielfaches des Abstands zwischen den beiden Mikrofonen.

Mit der Annahme, das Geräusch käme aus unendlicher Entfernung, würden Geraden, die durch die Geräuschquelle und die beiden Mikrofone verlaufen, parallel zueinander und zu der vorher erwähnten Geraden sein. Damit lässt sich ein Dreieck konstruieren, mit dessen Hilfe wir den Winkel α zur Geräuschquelle berechnen können[5] (siehe Abbildung 2.2).

$$\alpha = \frac{\pi}{2} - \arccos\left(\frac{\Delta t \ c}{d}\right) \tag{2.3}$$

Konstruieren wir das Dreieck mit dem eingezeichneten rechten Winkel, entspricht eine seiner Seitenlängen dem zeitlichen Versatz Δt, um den das Geräusch länger zu einem der Mikrofone braucht. Als dritte Größe kennen wir den Abstand d der Mikrofone zueinander. Bringt man physikalisch diesen Abstand und Δt auf dieselbe Einheit, indem man die Schallgeschwindigkeit c mit einfließen lässt, lassen sich alle weiteren Größen des Dreiecks berechnen. Auch der Winkel α zur Geräuschquelle(2.3), zu der sich der Roboter drehen soll, kann berechnet werden.

2.3 Kreuzkorrelation

Eine Kreuzkorrelation wird hier als eine einzelne Korrelation mit bestimmter Verschiebung betrachtet. Sie ist eine einfache Methode, um zu berechnen, wie ähnlich sich zwei Signale sind. Durch mehrere Kreuzkorrelationen der Signale vom linken und rechten Mikrofon mit unterschiedlichen Verschiebungen kann die zeitliche Verzögerung ermittelt werden, mit der ein Geräusch die beiden Mikrofone erreicht hat[6].

Audio-Signale, wie sie von Mikrofonen kommen, bestehen aus einer kontinuierlichen Folge von Samples. Bei einem Signal mit der Qualität einer Musik-CD beispielsweise sind es 44100 Samples pro Sekunde. Ein Sample hat einen Wert, welcher die Amplitude der Schwingung zu einem bestimmten Zeitpunkt abbildet. In einem Mikrofon schwingt eine Membran zum Schalldruck, die Amplitude der Schwingung entspricht dann der Stärke, mit der die Membran gerade in eine Richtung schwingt. Entsprechend dieser Richtung ist die Amplitude mit einem Vorzeichen behaftet.

Für eine Kreuzkorrelation werden zwei Signale mit einer festen Länge benötigt. Dazu wird eine bestimmte Anzahl von aufeinanderfolgenden Samples als eine Folge fester Länge zusammen gefasst. Eine solche Folge vom linken Mikrofon kann nun mit einer gleich langen Folge vom rechten Mikrofon durch eine Kreuzkorrelation verglichen werden.

$$K_{xy}(r) = \frac{1}{N - |r|} \sum_{n=0}^{N-1} x_n \ y_{n+r} \qquad -N < r < N \tag{2.4}$$

Mathematisch betrachtet handelt es sich bei der Kreuzkorrelation in dieser Arbeit um eine diskrete Kreuzkorrelation mit originalgetreuer Schätzung[7](2.4). Die beiden Folgen sind x und y, N ist die Länge einer Folge. Mit r wird eine Verschiebung der Folge y zur Folge x beschrieben, auf die später näher eingegangen wird.

Bei der Kreuzkorrelation wird jedes Sample der linken Folge mit dem Sample an derselben Stelle der rechten Folge multipliziert, also das Erste mit dem Ersten, das Zweite mit dem Zweiten und so weiter. Über die Produkte wird dann das arithmetische Mittel gebildet. Je höher das Ergebnis der Kreuzkorrelation ist, desto mehr gleichen sich die beiden Folgen und desto lauter waren sie, denn höhere Amplituden können höhere Ergebnisse bewirken. Da wir im Folgenden nur Kreuzkorrelationen derselben Folgen von Samples miteinander vergleichen

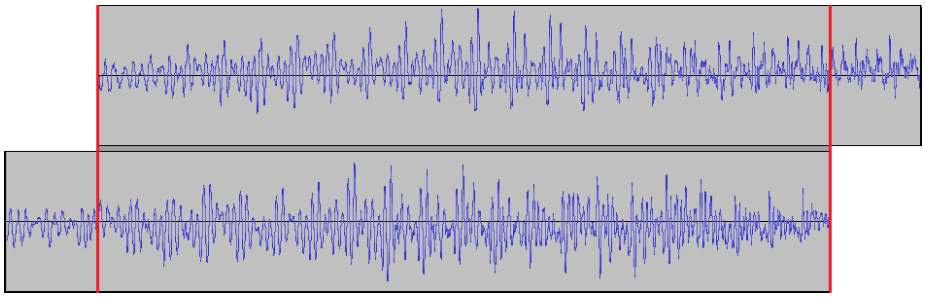

Abbildung 2.3: Zwei Folgen von Samples, oben vom linken und unten vom rechten Mikrofon, Folge vom Rechten ins zeitlich frühere verschoben, Samples zwischen den roten Linien werden multipliziert.

wollen, kürzt sich die Lautstärke gewissermaßen raus und hat keinen Einfluss. In der Theorie können auch negative Ergebnisse auftreten, wenn eine Folge der Inversen der anderen ähnelt, was in dem Anwendungsfall dieser Arbeit aber nicht vorkommt.

Die eben beschriebene Kreuzkorrelation hatte keine Verschiebung zwischen den beiden Folgen. Es ist aber auch möglich die Kreuzkorrelation mit zueinander verschobenen Folgen durchzuführen (siehe Abbildung 2.3). Beispielweise kann die Folge vom rechten Mikrofon im Verhältnis zur linken um ein Sample nach vorne verschoben werden, was einer zeitlichen Verschiebung um die Dauer eines Samples bedeutet. Bis zu einem Maximum, entsprechend der Längen der beiden Folgen, kann für jede Verschiebung die Kreuzkorrelation berechnet werden. Mit jedem Sample, um das man verschiebt, reduziert sich allerdings auch die Anzahl der multiplizierten Samples um eins, daher ist das Ergebnis, bei Verschiebungen in der Größenordnung der Anzahl der Samples, weniger aussagekräftig.

Bei der Verschiebung mit dem höchsten Ergebnis, passen die beiden Folgen am besten zueinander. Enthält diese Folge ein Geräusch, das mit seiner Lautstärke dominiert, dann ist die zeitliche Verzögerung, die sich aus dieser Verschiebung errechnen lässt, höchstwahrscheinlich die, mit der das Geräusch die beiden Mikrofone erreicht hat.

2.4 Korrelation im Frequenzraum

Eine Folge von Samples kann durch eine diskrete Fourier-Transformation (DFT) (2.5) in den Frequenzraum transformiert werden[8]. Dabei wird das Spektrum der zeitlichen Folge berechnet, welches aus Anteilen von Sinus- und Cosinusschwingungen mit unterschiedlichen Frequenzen besteht. Dieses Spektrum bildet eine Folge, die nun nicht mehr im Zeit- sondern im Frequenzraum liegt und komplex ist. Jeder Wert der Folge hat also einen Real- und einen Imaginärteil.

$$\underline{X}_k = \sum_{n=0}^{N-1} x_n \; e^{-i2\pi k \frac{n}{N}} \qquad k = 0, \ldots, N-1 \tag{2.5}$$

Die Folge beschreibt zu jeder diskreten Frequenz k ein $\underline{X}$ in der komplexen Ebene (siehe Abbildung 2.4). Dies wird deutlich, wenn man die Formel der DFT(2.5) umstellt(2.6).

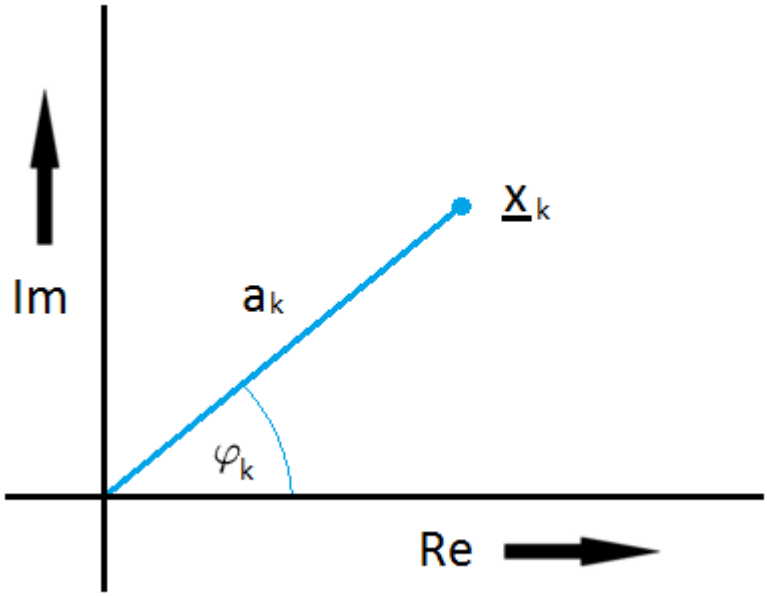

Abbildung 2.4: Geometrisches Verhältnis von $\underline{X}_k$, a_k und φ_k.

$$\underline{X}_k = \sum_{n=0}^{N-1} x_n \; e^{-i2\pi k \frac{n}{N}} = \sum_{n=0}^{N-1} x_n \left(\underbrace{\cos\left(2\pi k \frac{n}{N}\right)}_{\text{Realtteil}} - i \underbrace{\sin\left(2\pi k \frac{n}{N}\right)}_{\text{Imaginärteil}} \right) \tag{2.6}$$

Die höchste Frequenz im Frequenzraum ergibt sich aus der Abtastrate, also der Anzahl der aufgenommenen Samples pro Sekunde. Bei der bereits erwähnten Qualität einer Musik-CD mit 44100 Samples pro Sekunde ist die halb so hohe Frequenz von 22050 Hz gerade nicht mehr enthalten. Dies ergibt sich aus dem Nyquist-Shannon-Abtasttheorem[9]. Die Auflösung im Frequenzraum wird bestimmt durch die Länge der transformierten Folge.

$$f_k = k \; \frac{f_A}{N} \qquad k = 0, \ldots, N-1 \tag{2.7}$$

In der Formel zur Berechnung der Frequenz f_k(2.7), die vom k-ten Wert der transformierten Folge repräsentiert wird, steht f_A für die Abtastfrequenz und N für die Anzahl an Samples in der Folge. Obwohl die höchste Frequenz wie zuvor erwähnt auf unter 22050 Hz begrenzt ist, reicht f_k von 0 bis 44100 Hz. Ab der maximalen Frequenz, genau in der Mitte der Folge, wiederholt sich jedoch das Spektrum in gespiegelter Form, die höheren Frequenzen werden also nicht mit abgebildet.

Die Abtastrate einer Musik-CD liegt bei 44100 Samples pro Sekunde und wird in dieser Arbeit verwendet, da sie der menschlichen Wahrnehmung ent-

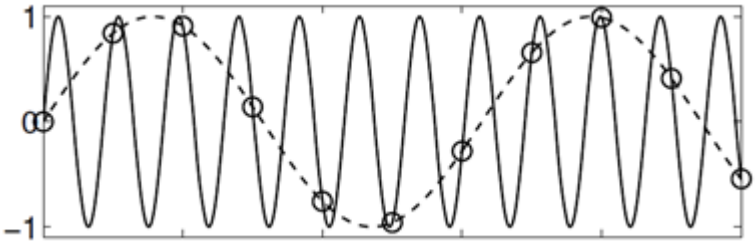

Abbildung 2.5: Aliasing-Effekt durch eine zu niedrige Abtastfrequenz, Kreise: Abtastpunkte, gestrichelt: Resultierendes falsches Signal[11].

spricht. Ein Mensch kann Frequenzen bis zu maximal 20 kHz wahrnehmen, wobei dieser Wert mit dem Alter sinkt. Um das Maximum an hörbarer Qualität in Bezug auf Frequenzen zu erlangen, wollte man mit Musik-CDs leicht darüber liegen. Man entschied sich letztlich für Frequenzen bis ausgenommen 22050 Hz durch eine Abtastrate von 44100 Hz, da eine Musik-CD so genug Speicherplatz für 74 Minuten bietet, was für die meisten Werke von Mozart und Beethoven gerade ausreichend ist[10].

Versucht man bei 44100 Samples pro Sekunden ein Signal mit einer Frequenz von 22050 Hz oder mehr abzutasten, kommt es zu einem Aliasing-Effekt (siehe Abbildung 2.5). Dabei wird eine falsche niedrigere Frequenz erkannt, die im abgetasteten Signal nicht vorhanden ist.

Man kann nun die Folgen vom linken und rechten Mikrofon in den Frequenzraum transformieren, zuvor werden in dieser Arbeit jedoch ihre Längen auf das Doppelte verlängert, indem Samples mit dem Wert Null angehängt werden. Wozu dies nötig ist, wird später erläutert. Von diesen beiden Folgen kann eine korrelierte Folge berechnet werden(2.8). Dies ist der gleiche Vorgang wie bei der Berechnung einer Kreuzkorrelation, ohne dass der arithmetische Mittelwert gebildet wird. Die Folge, die im Verhältnis zur anderen verschoben werden soll, also in dieser Arbeit die zweite Folge $\underline{Y}$, muss dabei konjugiert komplex sein. Das heißt, ihr Imaginärteil erhält ein negatives Vorzeichen.

$$\underline{C}_{xy}(k) = \underline{X}_k \; \underline{Y}_k^* \qquad k = 0, \ldots, 2\; N - 1 \tag{2.8}$$

Bei der Multiplikation einer komplexen mit einer konjugiert komplexen Zahl ergeben sich Real- und Imaginärteil des Ergebnisses entsprechend folgender Formeln(2.9)(2.10).

$$Re(\underline{X}\; \underline{Y}^*) = Re(\underline{X})\; Re(\underline{Y}) + Im(\underline{X})\; Im(\underline{Y}) \tag{2.9}$$

$$Im(\underline{X}\; \underline{Y}^*) = Im(\underline{X})\; Re(\underline{Y}) - Re(\underline{X})\; Im(\underline{Y}) \tag{2.10}$$

Die so korrelierte Folge kann mit der Inversen DFT(2.11) wieder vom Frequenzraum in den ursprünglichen Zeitraum zurück transformiert werden.

$$C_{xy}(r) = \frac{1}{N} \sum_{k=0}^{N-1} \underline{X_k}\; e^{+i2\pi k \frac{r}{N}} \qquad r = 0, \ldots, N - 1 \tag{2.11}$$

N entspricht der Länge der Folge, die zurück transformiert werden soll. Da wir die Länge durch das Anhängen von Nullen verdoppelt haben, ist es also doppelt so hoch, wie bei dem ursprünglichen Signal.

Diese Folge $C_{xy}(r)$ steht zu dem aus der Kreuzkorrelation bekannten $K_{xy}(r)$ in folgender Beziehung(2.12).

$$K_{xy}(r) = \frac{1}{N - |r|} \, C_{xy}(N + r) \qquad -N < r < N \tag{2.12}$$

N ist dabei die Anzahl der Samples in den Folgen, bevor die Nullen angehängt wurden. Dadurch, dass die Nullen den Folgen zu Anfang hinzugefügt wurden, hat die sich daraus ergebende Folge genug Elemente, um alle Verschiebungen r zu enthalten, denn r reicht von $-(N-1)$ bis $N-1$. Wie man sehen kann, ist es ausreichend $N-1$ Nullen anzuhängen, durch das Anhängen von N Nullen bleibt die Länge einer Folge jedoch eine Zweierpotenz, wenn sie es vorher war.

Das Ergebnis dieser Vorgehensweise ist also identisch mit dem, was bei der Kreuzkorrelation erreicht wird. Der Unterschied liegt darin, dass bei der Korrelation im Frequenzraum die Korrelationen für alle Verschiebungen errechnet werden, während eine Kreuzkorrelation immer nur für eine bestimmte Verschiebung errechnet wird. Verwendet man für die DFT statt der Formel(2.5) mit quadratisch vielen Rechenschritten, eine schnelle Fourier-Transformation (FFT)[5], können im Vergleich zur Berechnung aller Kreuzkorrelationen viele Rechenschritte eingespart werden. Dieser Vorteil gilt allerdings nur, wenn man tatsächlich alle Verschiebungen und nicht nur einige wenige berechnen möchte.

Ein weiterer Vorteil der Korrelation im Frequenzraum liegt in der Möglichkeit spezielle Filter auf die Folgen anzuwenden, die nur im Frequenzraum möglich sind. Konkret sind das in dieser Arbeit das Phase Only Matched Filter (Kapitel 2.5.3) und das Frequenzen lernende Filter (Kapitel 2.5.4).

2.5 Signalfilter

Signalfilter sind dazu da, Bestandteile eines Signals stärker hervorzuheben. In dieser Arbeit sollen interessante Geräusche und ihre Verschiebung hervorgehoben werden, während andere gleichzeitig aufgezeichnete Geräusche sowie Rauschen soweit wie möglich entfernt werden sollen.

2.5.1 Schwellwertfilter

Ein Schwellwertfilter filtert Werte heraus, die einen bestimmten Schwellwert nicht erreichen. In dieser Arbeit bedeutet dies, dass in den Signalen vom linken und rechten Mikrofon alle Samples auf Null gesetzt werden, deren Amplituden den Noise-Level (Kapitel 2.1) nicht überschreiten(2.13).

$$x_{n\ \text{neu}} = \begin{cases} x_n & \text{falls } |x_n| > v \\ 0 & \text{sonst} \end{cases} \qquad n = 0, \ldots, N-1 \tag{2.13}$$

Da die Amplituden in positiver wie auch negativer Richtung ausschlagen, bildet der Noise-Level v einen Schwellwert gegen den Betrag der Amplitude des aktuellen Samples x_n. Anstatt, dass die Amplitude größer sein muss, könnte sie auch größer-gleich sein. Dies würde im Ergebnis allerdings kaum etwas ändern

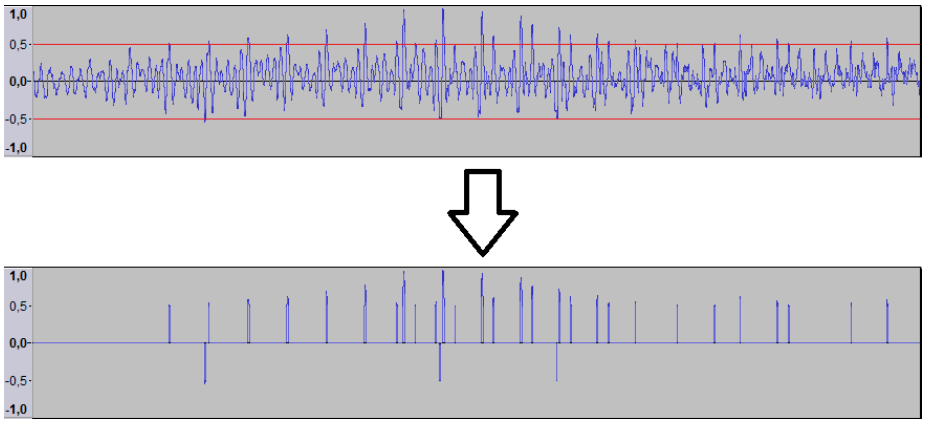

Abbildung 2.6: Oben: Signal und Noise-Level in rot, unten: Signal nach Anwendung des Schwellwertfilters.

und so ist das Schwellwertfilter besser vergleichbar mit dem Subtraktionsfilter (Kapitel 2.5.2).

Mathematisch betrachtet ist dies eine nicht lineare Operation, was starke Veränderungen im Spektrum (Kapitel 2.4) mit sich führt, da Sprünge eingefügt werden.

Bei diesem Filter gehen sehr viele Informationen des Signals verloren, es bleiben nur noch die Spitzen der höchsten Amplituden übrig (siehe Abbildung 2.6). Selbst wenn das Signal ausschließlich das gewollte interessante Geräusch enthalten würde und keine Störeinflüsse, würde der größte Teil der Amplituden verloren gehen. Auf diesen Teil kann aber möglicherweise verzichtet werden, denn was hier letztlich errechnet werden soll, ist eine eindeutigere Verschiebung dieses Signals zu einem anderen. Werden beide Signale von diesem Filter verarbeitet, lassen sich die Spitzen potenziell einfacher und eindeutiger zuordnen als die beiden ungefilterten Signale.

Der Grund, warum diese Spitzen in den Amplituden so aussagekräftig sind, liegt darin, dass dort der Anteil der Störgeräusche tendenziell am geringsten ist. Bei Rauschen, Vibrationen des Roboters selbst und auch anderen denkbaren Störgeräusche kann angenommen werden, dass sie zu jedem Zeitpunkt etwa gleich stark wirken. Dies bedeutet, dass eine niedrige Amplitude fast ausschließlich durch Störgeräusche entstanden sein könnte, während mit steigender Höhe der Amplitude oberhalb des Noise-Levels der Anteil eines interessanten Geräusches an der Amplitude zunimmt.

Der Einsatz von einem solchen Schwellwertfilter in der Verarbeitung von Audiosignalen ist ungewöhnlich, weil dabei so viele Informationen verloren gehen und erst ein sinnvoller Schwellwert gefunden werden muss. In diesem speziellen Fall, der in dieser Arbeit behandelt wird, bietet er sich jedoch an, da nur die Verschiebung der Signale von Interesse ist und ein sinnvoller Schwellwert durch den Noise-Level bereits gegeben ist.

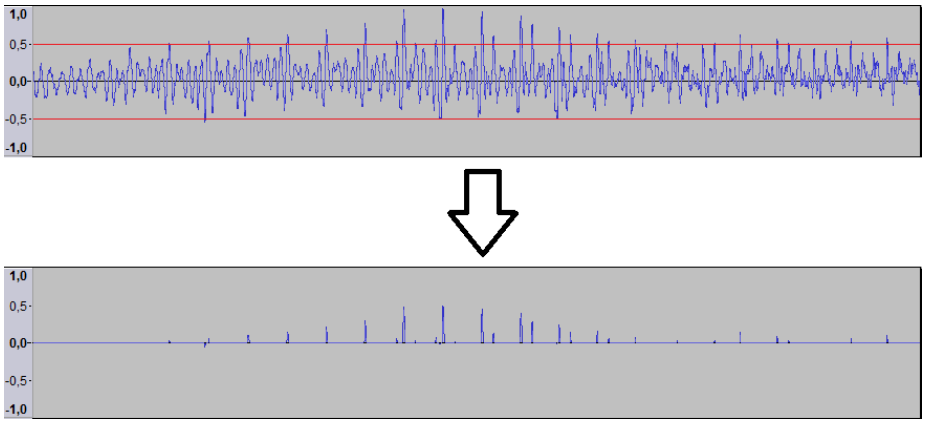

Abbildung 2.7: Oben: Signal und Noise-Level in rot, unten: Signal nach Anwendung des Substraktionsfilters.

2.5.2 Subtraktionsfilter

Das Subtraktionsfilter ähnelt stark dem Schwellwertfilter (Kapitel 2.5.2), wobei zusätzlich der Schwellwert von dem Betrag der Amplituden subtrahiert wird, wenn sie nicht auf Null gesetzt werden. Anders ausgedrückt wird der Schwellwert von dem Betrag aller Amblituden subtrahiert, wobei die Null nicht unterschritten werden kann(2.14).

$$x_{n\ \text{neu}} = \begin{cases} x_n - v & \text{falls } x_n > v \\ x_n + v & \text{falls } x_n < -v \\ 0 & \text{sonst} \end{cases} \qquad n = 0, \ldots, N-1 \tag{2.14}$$

Nach diesem Vorgang enthält das Resultat exakt dieselben Informationen wie bei dem Schwellwertfilter. Bei dem Subtraktionsfilter ist der Unterschied in den verbleibenden Amplituden über Null jedoch im Verhältnis zu ihrer Größe höher. Eine Amplitude, die deutlich über dem Schwellwert liegt, kann nach Anwendung des Filters um ein vielfaches höher sein als eine Amplitude, die den Schwellwert nur knapp überragt hat (siehe Abbildung 2.7).

Durch das Subtraktionsfilter wirken sich die Höhenunterschiede der Amplituden oberhalb des Noise-Levels stärker aus, was zu einer eindeutigeren Zuordnung führen kann.

Bei den Recherchen zu dieser Arbeit konnte kein Beispiel der Anwendung eines solchen Filters gefunden werden. Es ist davon auszugehen, dass dies ein sehr ungewöhnliches Vorgehen ist, dessen Resultate bisher kaum mit denen herkömmlicher Methoden verglichen wurden. Ein Subtraktionsfilter bietet sich dennoch an, aus denselben Gründen wie schon bei dem Schwellwertfilter (Kapitel 2.5.2).

2.5.3 Phase Only Matched Filter

Bei dem Phase Only Matched Filter (POMF)[1] handelt es sich um ein Filter, das ein Signal nicht im ursprünglichen Zeitraum, sondern im Frequenzraum

manipuliert. Jedes Sample im Frequenzraum ist eine komplexe Zahl, enthält also einen Real- und einen Imaginärteil. Daneben gibt es jedoch noch weitere Werte, die sich daraus berechnen lassen (siehe Abbildung 2.4).

Real- und Imaginärteil des Wertes eines Samples $\underline{X}_k$ bilden einen Vektor, dessen euklidische Länge a_k ist und Amplitude genannt wird(2.15), was nicht verwechselt werden darf mit dem Begriff der Amplitude im Zeitraum. Der Winkel φ_k des Vektors wird als Phase bezeichnet(2.16).

$$a_k = \sqrt{Re(\underline{X}_k)^2 + Im(\underline{X}_k)^2} \qquad k = 0, \ldots, N-1 \tag{2.15}$$

$$\varphi_k = \arctan 2 \left(\frac{Im(\underline{X}_k)}{Re(\underline{X}_k)} \right) \qquad k = 0, \ldots, N-1 \tag{2.16}$$

Das Phase Only Matched Filter macht sich zunutze, dass die Information über die Verschiebung zweier Signale zueinander ausschließlich in ihren Phasen enthalten ist[1]. Entscheidend ist also nicht die Höhe von Real- und Imaginärteil, sondern das Verhältnis zwischen diesen beiden Größen. Nach einer Korrelation werden Samples jedoch abhängig von der Höhe ihrer Werte unterschiedlich stark gewichtet. POMF entfernt diese Gewichtung, sodass nur noch die Phasen aller Samples das Ergebnis beeinflussen.

Dies wird erreicht, indem die Amplituden aller Samples im Frequenzraum auf den Wert Eins normalisiert werden. Daraus ergeben sich folgende Formeln für den neuen Real-(2.17) und Imaginärteil(2.18) des gefilterten Signals.

$$Re(\underline{X}_{k\ \mathrm{neu}}) = \cos(\varphi_k) \qquad k = 0, \ldots, N-1 \tag{2.17}$$

$$Im(\underline{X}_{k\ \mathrm{neu}}) = \sin(\varphi_k) \qquad k = 0, \ldots, N-1 \tag{2.18}$$

Zu beachten ist, dass diese Formeln nur angewendet werden können, wenn der Realteil von $\underline{X}_k$ ungleich Null ist, denn bei der Berechnung von φ_k wird durch ihn geteilt. In diesem Fall gibt es kein definiertes φ_k. Bei Samples, auf die dies zutrifft, werden Real- und Imaginärteil zu Null. Dieser Fall tritt besonders häufig auf, wenn vorher auf dem Signal im Zeitraum das Schwellwert- (Kapitel 2.5.1) oder Subtraktionsfilter (Kapitel 2.5.2) angewendet wurde, da diese viele Nullen hinterlassen.

Da es sich um die Normalisierung eines Vektors handelt, kann das gleiche Ergebnis auch durch eine kürzere Formel(2.19) erreicht werden.

$$\underline{X}_{\mathrm{k\ neu}} = \begin{cases} \frac{\underline{X}_k}{a_k} & \text{falls } a_k \neq 0 \\ 0 & \text{sonst} \end{cases} \qquad k = 0, \ldots, N-1 \tag{2.19}$$

Des Weiteren kann dieses Filter auch nach der Korrelation im Frequenzraum und bevor das Signal wieder in den Zeitraum transformiert wurde angewendet werden. Dies bedeutet, dass das POMF statt auf beiden Signalen vom linken und rechten Mikrofon, nur einmal auf das korrelierte Signal angewendet werden braucht, was viele Rechenschritte spart.

Versuche aus anderen Arbeiten[12] haben gezeigt, dass POMF im Ergebnis Verschiebungen mit geringerem Fehler liefern kann. Allerdings wird die unterschiedliche Lautstärke von Geräuschen im Signal dabei ignoriert.

2.5.4 Frequenzen lernendes Filter

Das Frequenzen lernende Filter ist ein Konzept, das im Rahmen dieser Arbeit entstanden ist. Es gehört in die Kategorie der adaptiven Filter[13] und stellt eine Art lernendes Wiener Filter[14] dar. Das in diesem Kapitel beschriebene Filter arbeitet im Frequenzraum auf dem Signal und nutzt aus, dass die Amplituden dort für die Stärke, also auch die Lautstärke stehen, mit der bestimmte Frequenzen vorkommen. Es lernt, wie stark jede erfasste Frequenz in der Vergangenheit vertreten war und senkt abhängig davon deren Amplituden, um ihren Einfluss auf das Ergebnis der Korrelation zu senken. Das Prinzip ist von der sogenannten Kurz-Zeit-Habituation[15] des Menschen inspiriert, der monotone Hintergrundgeräusche, wie beispielsweise die von einem Lüfter, nach einer gewissen Zeit automatisch ausblendet beziehungsweise auf den Reiz schwächer reagiert.

Die Amplituden der Frequenzen können genauso gelernt werden wie der Noise-Level (Kapitel 2.1) , der sich ebenfalls über die Zeit anpasst. Das Prinzip ist das Gleiche, jedoch findet der Lernprozess für jede einzelne Frequenz f_k(2.7) unabhängig voneinander statt.

$$a_{\text{average}} = \frac{1}{N} \sum_{k=0}^{N-1} a_k \qquad k = 0, \ldots, N-1 \tag{2.20}$$

$$l_{k,0} = \frac{a_k}{a_{\text{average}}} \qquad k = 0, \ldots, N-1 \tag{2.21}$$

$$l_{k,m} = l_{k,m-1}\ (1-\alpha) + \frac{a_k}{a_{\text{average}}}\ \alpha \qquad k = 0, \ldots, N-1 \tag{2.22}$$

Initial wird $l_{k,0}$ berechnet(2.21), indem die Amplitude der aktuell betrachteten Frequenz a_k durch das arithmetische Mittel der Amplituden über alle Frequenzen a_{average}(2.20) geteilt wird. Das Ergebnis ist also das Verhältnis der betrachteten Frequenz zum Durchschnitt.

Mit jeder Folgemessung wird der gelernte Wert $l_{k,m}$ angepasst(2.22). Die neue Messung wird dabei mit α gewichtet.

Um anhand des Gelernten eine Amplitude zu verringern, muss aus dem gelernten l_k ein Faktor zwischen Null und Eins berechnet werden, wobei Null die Amplitude einer Frequenz vollständig verschwinden lassen würde, während Eins sie unverändert lässt. Dies wird in der folgenden Formel(2.23) erreicht.

$$\underline{X}_{k\ \text{neu}} = \underline{X}_k\ \max\left\{0,\ 1 - \frac{l_k}{L}\right\} \qquad k = 0, \ldots, N-1 \tag{2.23}$$

Um die Amplitude um einen Faktor zu verringern, kann die komplexe Zahl $\underline{X}_k$, aus der sie sich berechnet, direkt mit dem Faktor multipliziert werden. Der gelernte Wert l_k wird durch eine Konstante L geteilt, diese bestimmt, wie hoch eine gelernte Amplitude im Verhältnis zum Durchschnitt sein darf, bis der sich daraus ergebende Faktor auf Null sinkt. Das Ergebnis der Division ist eine positive Zahl, je höher, desto stärker war die Frequenz n in der Vergangenheit vertreten und desto niedriger soll der Faktor sein. Darum wird dieser Wert von Eins subtrahiert, die Funktion max mit dem Element Null stellt sicher, dass der Faktor nicht negativ wird, falls $l_k > L$.

Daraus ergibt sich beispielsweise, dass bei einem $L = 10$ die Amplitude einer Frequenz, die gelernt doppelt so hoch ist wie der Durchschnitt, mit dem Faktor

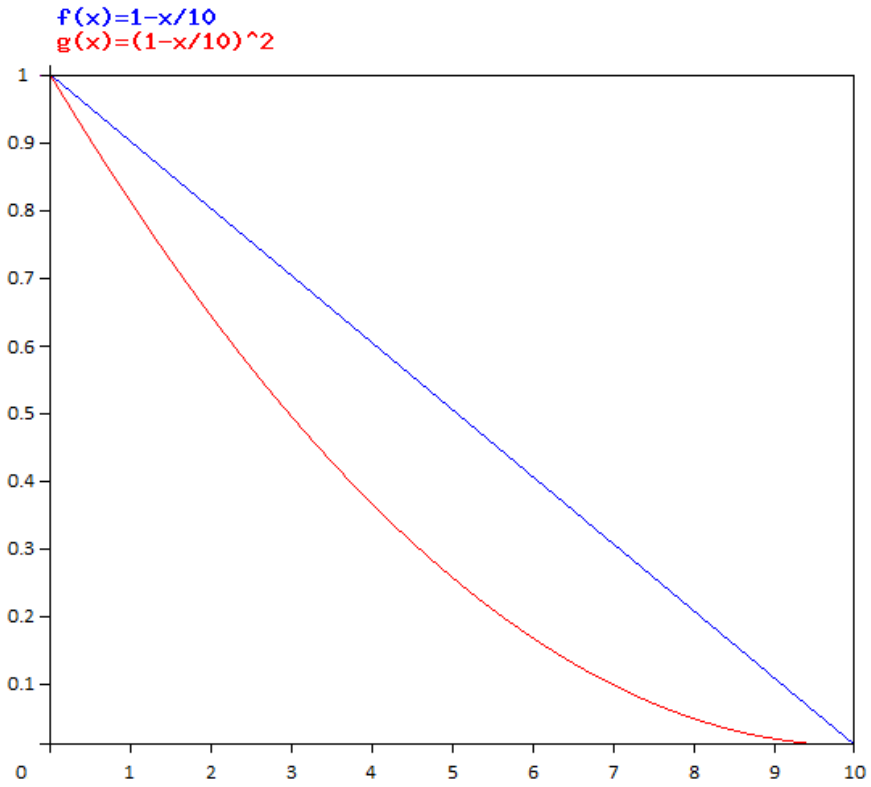

Abbildung 2.8: Blau: Faktor der ersten Version der Formel, rot: zweite Version mit quadriertem Faktor. X-Achse: Gelernte Amplitude im Verhältnis zum Durchschnitt, Y-Achse: Errechneter Faktor.

0,8 multipliziert wird. Ist sie nur halb so hoch, beträgt der Faktor 0,95. Ab einer gelernten Amplitude, die L-mal, also zehnmal höher ist als der Durchschnitt, sinkt der Faktor auf Null.

Eine zweite Variante der Formel, zur Anwendung der gelernten Amplitude, quadriert zusätzlich den Faktor(2.24).

$$\underline{X}_{k\ \text{neu}} = \underline{X}_k \left(\max \left\{ 0, 1 - \frac{l_k}{L} \right\} \right)^2 \qquad k = 0, \ldots, N-1 \tag{2.24}$$

Der daraus entstehende Verlauf des Faktors, bei steigender gelernter Amplitude im Verhältnis zum Durchschnitt, kann das Ausblenden von Frequenzen möglicherweise natürlicher wiedergeben (siehe Abbildung 2.8).

Wie schon beim Phase Only Matched Filter (Kapitel 2.5.3) ist es für die Anzahl an notwendigen Rechenschritten von Vorteil das Frequenzen lernende Filter auf das Signal anzuwenden, das bereits korreliert ist, aber noch nicht in den Zeitraum zurück transformiert wurde. Dies ist möglich, allerdings ändert sich dadurch das Ergebnis.

Da im korrelierten Signal die beiden Signale vom linken und rechten Mikrofon miteinander multipliziert wurden, werden die Unterschiede zwischen den

Amplituden tendenziell größer. Wären beide ursprünglichen Signale identisch, würde jede Amplitude quadriert werden, dementsprechend müsste auch die Konstante L quadriert werden. Dem wirkt ausgleichend entgegen, dass auch das arithmetische Mittel über alle Amplituden steigt, jedoch nicht im gleichen Maße.

In der Praxis kann das Frequenzen lernende Filter auch auf das korrelierte Signal angewendet werden, da sich die Signale von zwei Mikrofonen in den aufgezeichneten Frequenz-Amplituden zwar unterscheiden, sich aber sehr ähnlich sein sollten, wodurch sich die Veränderungen an den Amplituden-Unterschieden und ihrem arithmetischen Mittel weitestgehend aufheben. In beiden Fällen muss ein praxistauglicher Wert für die Konstante L gefunden werden.

Die Wirkung des Frequenzen lernenden Filters ähnelt der des Phase Only Matched Filters. Bei diesem werden die Amplituden auf denselben Wert angeglichen. Beim Frequenzen lernenden Filter findet ebenfalls ein Angleichungseffekt statt, denn tendenziell werden hohe Amplituden stärker verringert, als niedrige. Diese Anpassung basiert jedoch auf vorhergehenden Messungen, sodass weniger häufig auftretende und somit interessantere Frequenzen weniger abgeschwächt werden, auch wenn sie hoch sind.

Kapitel 3

Implementierung

In den vorherigen Kapiteln wurden die Ziele dieser Arbeit beschrieben und welche Konzepte dazu genutzt werden sollen. In diesem Kapitel wird die konkrete Umsetzung der Konzepte erläutert. Dabei werden die Modifikationen an der Hardware des Roboters und die entstandene Software erklärt.

3.1 Verwendete Hardware

Der Roboter Doggy wird gesteuert von einem Mikrocontroller, auf dem alle seine bisherigen Fähigkeiten, also das Erkennen von Bällen, Berechnen der Flugbahnen und Zurückschlagen, implementiert sind. Zu Doggy gehört auch ein herkömmlicher Computer, der per USB mit dem steuernden Mikrocontroller verbunden wird. Dieser Computer war bisher ohne Funktion, übernimmt nun aber das Berechnen interessanter Geräusche, um die es in dieser Arbeit geht.

Damit Doggy stereo hören kann, braucht er zwei Mikrofone. Für diesen Zweck wurde im Rahmen dieser Arbeit nach einiger Recherche das Modell ME-52W von Olympus ausgewählt. Dabei handelt es sich um ein Ansteckmikrofon, das man normalerweise an eine Krawatte oder einen Hemdkragen ansteckt, um Sprache von einer Person aufzunehmen.

Mikrofone besserer Qualität sind deutlich größer und brauchen Phantomspeisung, was bedeutet, dass sie durch das Aufnahmekabel auch mit Strom versorgt werden müssen. Dazu muss ein zusätzliches versorgendes Gerät zwischengeschaltet werden, das ebenfalls in Doggy eingebaut werden müsste und auf einen zusätzlichen Stromanschluss oder aufgeladene Batterien angewiesen wäre. Zudem bewegen sich Mikrofone dieses Typs in einer weitaus höheren Preisklassen.

Das ausgewählte Mikrofon ist sehr klein und zeichnet mono, also nur einen Kanal auf. Zwei dieser Mikrofone wurden links und rechts in Doggys Körper befestigt, um so stereo aufzuzeichnen (siehe Abbildung 3.1). Es wurden mehrere Befestigungsmöglichkeiten der Mikrofone mit Gummibändern getestet. Ziel war es, dass Vibrationen des Roboters sich nur möglichst schwach über das Gestänge auf die Mikrofone übertragen. Am Ende stellte sich eine sehr simple Variante als die Beste heraus, bei der die Mikrofone einfach an ihren Kabeln hängen. Der Abstand zwischen den Mikrofonen beträgt 24cm.

Die Mikrofone verfügen laut Beschreibung über eine nierenförmige Richtcha-

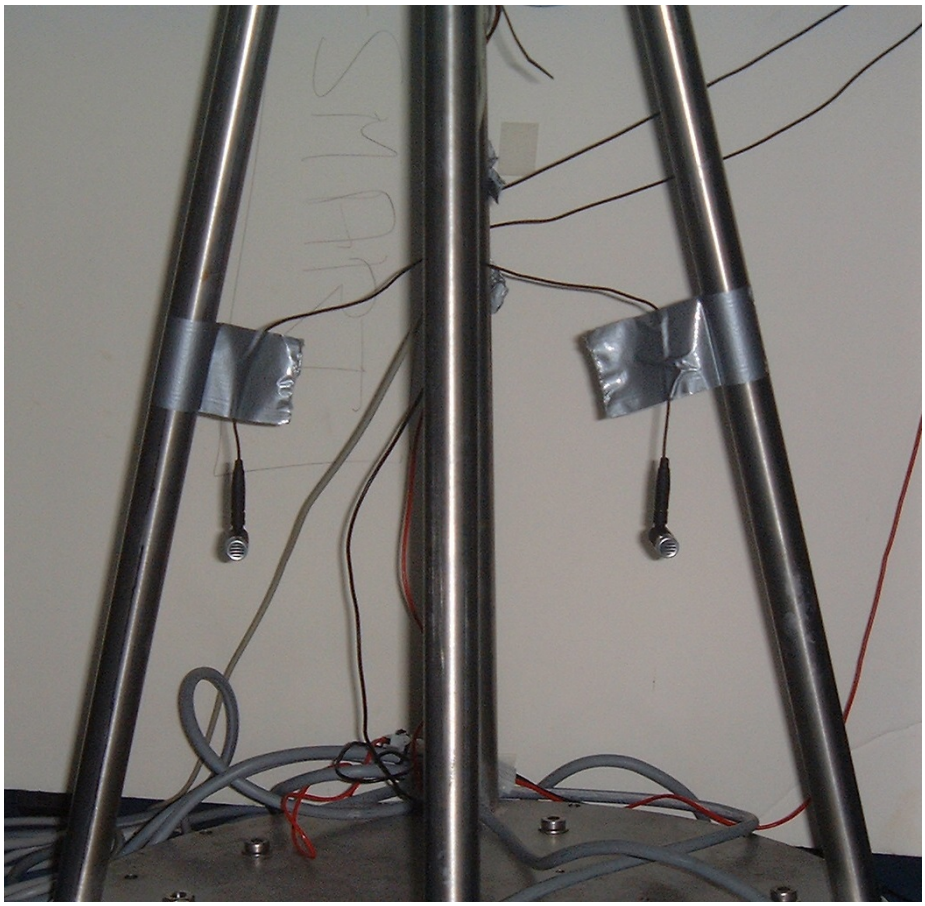

Abbildung 3.1: Mikrofone im Körper von Doggy montiert, sodass sich Vibrationen möglichst schwach auf sie übertragen.

rakteristik, es konnte bei Tests aber nur eine minimale Abnahme der Lautstärke bei Geräuschen von hinten festgestellt werden. Versuche die Mikrofone mit Schaumstoff von hinten gegen Geräusche abzuschirmen, blieben ohne spürbare Wirkung und wurden daher wieder verworfen.

Über ein Y-Kabel werden die beiden Monosignale zu einem Stereosignal, das als solches direkt an den Computer angeschlossen werden kann. Das Mainboard des Computers ist ein ASUS P8Z68V S1155 Z68 ATX mit einem leistungsfähigen RealTek ALC892 Soundchip, der über einen entsprechenden Stereoeingang verfügt und eine Sample Rate von bis zu 192kHz ermöglicht.

Um in Echtzeit Berechnungen für den Roboter durchführen zu können, ist der Computer mit einem Intel®CORE I7-2600k Prozessor ausgestattet.

Doggys Gestänge ist so gebaut, dass der Computer in seinem Körper untergebracht werden kann, jedoch nur in einer schrägen Position, die eines der Mikrofone verdeckt (siehe Abbildung 3.2). Während der Versuche zu dieser Arbeit befand sich der Computer daher nicht zwischen dem Gestänge, sondern blieb außerhalb des Roboters. Eine dauerhafte Lösung könnte durch ein angepasstes Gestänge in Doggys Nachfolger erreicht werden oder durch die Verwendung

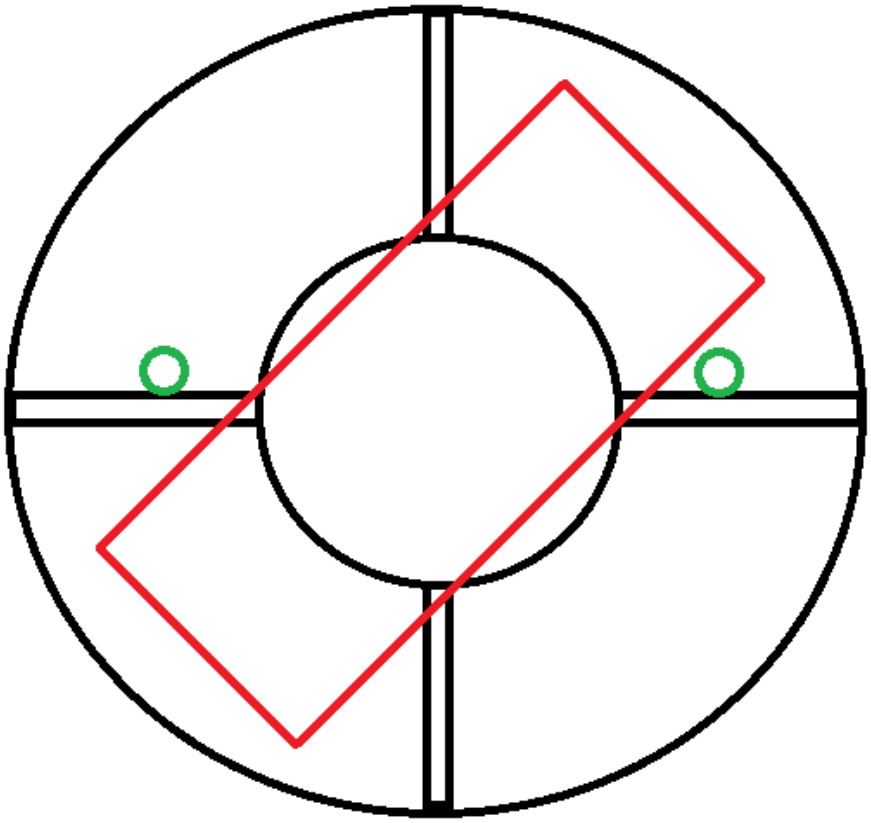

Abbildung 3.2: Schematische Ansicht des Roboters von oben, grüne Kreise: Mikrofone am Gestänge befestigt, rotes Rechteck: Position des Computers im Roboter.

eines kleineren Computers.

Alternativ hätte man auch versuchen können die Mikrofone oberhalb des Gestänges zu befestigen. Das gesamte Oberteil des Roboters soll sich zu den Geräuschen drehen und so würden sich die Mikrofone selbst mitdrehen. Dies würde zu einem Mitdrehen des Öffnungswinkels der Mikrofone führen, der höchstens 180° betragen kann (Kapitel 2.2). Bisher wurde der Roboter immer so eingesetzt, dass Bälle nur aus einer Richtung auf ihn geworfen wurden, daher sollte Doggy sich im Rahmen dieser Arbeit auf eine feste Richtung konzentrieren und die Mikrofone nicht mitdrehen lassen. Da aber auch die Kameras des Roboters sich mitdrehen und bei einem tatsächlichen Spiel „Schweinchen in der Mitte" Bälle aus allen Richtungen kommen können (Kapitel 1.1), wäre ein Mitdrehen der Kameras für zukünftige Versionen eine Möglichkeit.

3.2 Einsatz im ROS

Das Robot Operating System (ROS) bietet ein Framework, das es Forschern erleichtern soll Software für Roboter zu entwickeln. Wenn man ROS auf einem Computer startet, kann es mehrere Prozesse zur Laufzeit überwachen und miteinander kommunizieren lassen. Diese Prozesse werden ROS-Knoten genannt und damit sie in ROS laufen können, müssen sie eine bestimmte Art der Kommunikation über Sockets implementiert haben. Diese wird für mehrere Programmiersprachen bereits durch Bibliotheken von ROS frei zur Verfügung gestellt.

Der Vorteil von ROS liegt in der Wiederverwendbarkeit der ROS-Knoten. Soll eine Fähigkeit für Roboter neu entwickelt oder verbessert werden, sind dafür in der Regel andere Fähigkeiten des Roboters notwendig. So ist zur Wegfindung beispielsweise Hinderniserkennung eine Voraussetzung. Für alle Fähigkeiten des Roboters, die zwar notwendig sind, aber nicht den eigentlichen Kern einer Arbeit darstellen, können bereits vorhandene ROS-Knoten verwendet werden. Diese ROS-Knoten könnten an verschiedenen Enden der Welt in unterschiedlichen Programmiersprachen geschrieben worden sein und dennoch könnten sie in ROS gemeinsam laufen und miteinander kommunizieren.

Ein Grund für die hohe Wiederverwendbarkeit liegt in der Art, wie ROS-Knoten kommunizieren: Nach einem Publish-Subscriber-Model. Sie schicken Informationen nicht gezielt an andere Knoten, sondern publizieren ihre Ergebnisse unter einem bestimmten Namen als Nachricht in das laufende ROS. Andere ROS-Knoten können nach Nachrichten mit diesem Namen lauschen und sie so erhalten. Ein ROS-Knoten kümmert sich dabei nicht darum, welche, ob oder wie viele andere Knoten seine Nachrichten lesen. Ebenso ist es für einen ROS-Knoten auch nicht von Interesse, ob noch andere dieselben Nachrichten lesen. Ein ROS-Knoten liest einfach Nachrichten, die Informationen enthalten, die er braucht und publiziert seine eigenen Ergebnisse als neue Nachrichten. So können unterschiedlichste Knoten gleichzeitig im ROS laufen und gemeinsam arbeiten, solange sie nur dieselben Nachrichten verwenden.

Doggy besaß bereits alle seine bisherigen Fähigkeiten ohne ROS. Mit Blick auf mögliche Erweiterungen in der Zukunft wurde nun im Rahmen dieser Arbeit ROS auf Doggys Computer eingerichtet. Die gesamte Software läuft in ROS, aufgeteilt in vier Knoten: Perception (Kapitel 3.3), Model (Kapitel 3.4), Motion (Kapitel 3.5) und Control (Kapitel 3.6).

Es wäre hilfreich gewesen einen bereits vorhandenen ROS-Knoten zu verwenden, der die Signale von den Mikrofonen ins ROS publiziert. Es ist zum Zeitpunkt dieser Arbeit jedoch nur ein derartiger ROS-Knoten veröffentlicht und er publiziert die Signale mit verlustbehafteter MP3-Kodierung. Somit wurde kein bereits veröffentlichter ROS-Knoten verwendet.

Die Knoten kommunizieren über vier Nachrichten-Typen: SOI Percept, Modelled SOI Percept, Joints Request und Joints (siehe Abbildung 3.3). Jeder Nachrichten-Typ wird von einem Knoten erzeugt und von einem darauffolgenden gelesen, ausgenommen der Nachrichten-Typ Joints, der bisher von keinem Knoten beachtet wird. Wie zuvor erläutert wurde, könnten in Zukunft jedoch auch andere Knoten diese Nachrichten lesen oder publizieren.

Die vier ROS-Knoten, die im Rahmen dieser Arbeit entstanden sind, wurden alle in C++ implementiert. Gemeinsam mit ihren zugehörigen Nachrichtentypen sind sie in zwei Pakete unterteilt: SOI und RoboControl. SOI ist die Abkürzung für Sound Of Interest und enthält alles zum Erkennen von inter-

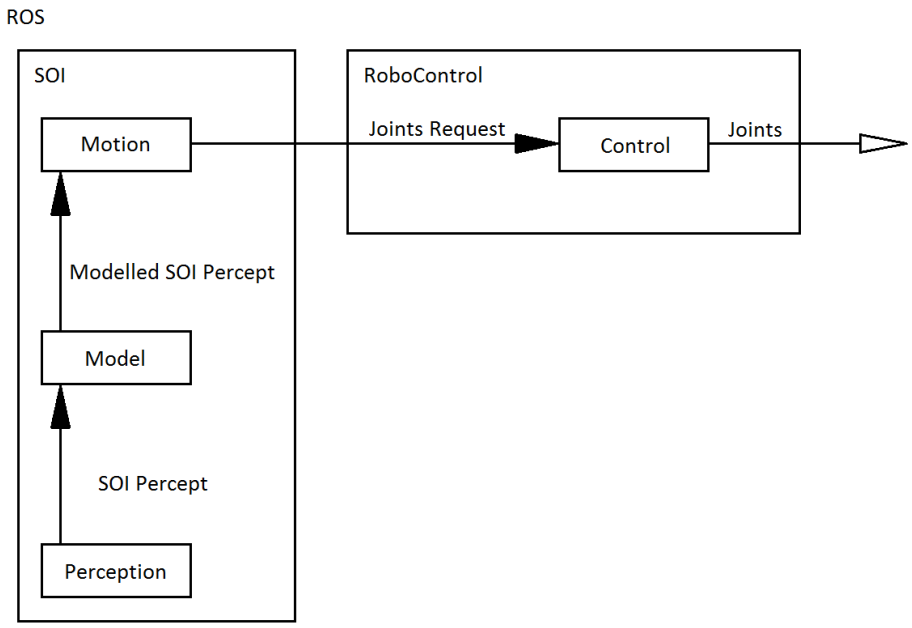

Abbildung 3.3: Architektur der entstandenen Knoten in ROS (kleine Kästen) und ihrer Nachrichtentypen (Pfeile), aufgeteilt in zwei Pakete (große Kästen).

essanten Geräuschen. RoboControl bietet eine Schnittstelle zwischen ROS und Doggys Mikrocontroller.

Der dokumentierte Quellcode aller ROS-Knoten und Nachrichtentypen dieser Arbeit ist auf dem beiliegenden Datenträger enthalten.

3.3 Perception

Der ROS-Knoten Perception hat als einziger direkten Zugriff auf das Sound-Signal. Seine Aufgabe umfasst das wesentlichste dieser Arbeit, nämlich bei Überschreiten des Noise-Levels (Kapitel 2.1) eine Korrelation durchzuführen und so die Verschiebung zwischen den Signalen zu ermitteln. Zusätzlich wird eine Qualität der Verschiebung berechnet, die angibt, wie deutlich diese ist. Die Verschiebung und ihre Qualität werden in ein SOI Percept (Kapitel 3.2) geschrieben und so in das laufende ROS publiziert. Dabei ist zu beachten, dass dies noch kein erkanntes interessantes Geräusch ist, sondern erst nachfolgend im Model (Kapitel 3.4) zu einem werden kann. Wird der Noise-Level nicht überschritten, werden leere SOI Percepte publiziert.

Die Sound-Signale können aus zwei Quellen kommen. Wird Perception ohne Kommandozeilenparameter gestartet, versucht es ein Signal von angeschlossenen Mikrofonen zu erhalten. Erhält Perception einen Kommandozeilenparameter, wird dieser als Pfad zu einer Datei im WAV-Format interpretiert und statt von den Mikrofonen wird das Signal aus dieser Datei ausgelesen.

Perception erwartet ein 16bit Stereosoundsignal mit einer Sampling-Rate von 44100 Hz. Die Kodierung in 16bit ganzzahlig mit Vorzeichen ist die am weitesten verbreitete und bietet ausreichend Qualität, die Sampling-Rate entspricht der

einer Musik-CD, die auf das menschliche Hörvermögen abgestimmt ist.

Um eine Korrelation durchführen zu können, müssen die Sound-Samples in Folgen bestimmter Länge unterteilt werden. In dieser Arbeit werden Folgen mit 4096 Samples vom linken und ebenso vielen vom rechten Mikrofon verwendet. Bei dieser Zahl handelt es sich um 2^{12}, also eine Zweierpotenz, was für FFTs (Kapitel 2.4) nötig ist. Die Höhe dieser Zweierpotenz hat sich in frühen Tests im Rahmen dieser Arbeit als brauchbar erwiesen.

Es wäre möglich, das kontinuierliche Stereosoundsignal in Folgen zu zerteilen und diese sequenziell zu korrelieren, jedoch würden dann Samples, die am Rand einer Folge liegen, oft nicht zu der Korrelation beitragen. Angenommen ein Geräusch würde nur aus einem einzelnen kurzen Ausschlag bestehen und hat das linke Mikrofon früher erreicht als das rechte, dann könnte dieses Geräusch gerade an der Grenzen zwischen zwei Folgen liegen. Damit läge es am Ende einer linken Folge und am Anfang der zeitlich späteren rechten. Diese beiden würden nicht miteinander korreliert werden und das Geräusch wäre praktisch überhört worden.

Eine Lösung für dieses Problem, die hier angewendet wird, ist das Überlappen von Folgen. Die ersten 50% einer Folge sind hier immer die letzten 50% der vorherigen und nur die zweite Hälfte der neuen Folge ist wirklich neu. So ist jedes Sample zweifach in einer Korrelation beteiligt und liegt mindestens einmal davon nicht am Rand der Folge. Es wäre auch eine Überlappung von weniger als 50% möglich, dies würde aber dazu führen, dass manche Samples häufiger korreliert werden als andere. Eine Überlappung über 50% würde die Rechenzeit erhöhen und keine weiteren Vorteile bringen.

Aus der Sampling-Rate f_A, der Länge N einer Folge und der Überlappung q lässt sich die Anzahl an Folgen pro Sekunde $f_{\text{perception}}$ berechnen(3.1).

$$f_{\text{perception}} = \frac{f_A}{q\ N} = \frac{44100\,\text{Hz}}{0,5 * 4096} \approx 21,5\,\text{Hz} \tag{3.1}$$

Die Anzahl der Folgen pro Sekunde $f_{\text{perception}}$ ist auch die Anzahl publizierter SOI Percepte pro Sekunde, da auch für Folgen, in denen der Noise-Level nicht überschritten wird, ein leeres SOI Percept publiziert wird.

Zur Berechnung des Noise-Levels werden die Konstanten F und α benötigt (2.1). Frühe Tests haben ergeben, dass $F = 1.4$ und $\alpha = 0.01$ brauchbare Ergebnisse liefern, wobei α in Abhängigkeit zu $f_{\text{perception}}$ steht, wenn die Anpassungsgeschwindigkeit pro Sekunde gleich bleiben soll.

Für die Korrelation können Filter angewendet werden. Alle bei den Konzepten erwähnten Signalfilter (Kapitel 2.5) sind im Rahmen dieser Arbeit auch implementiert worden und stehen Perception zur Verfügung. Das Schwellwertfilter und Subtraktionsfilter können nicht zusammen verwendet werden, ebenso das Phase Only Matched Filter und Frequenzen lernende Filter. Aus dieser Einschränkung ergeben sich neun mögliche Kombinationen von Filtern, eines von der linken Seite mit einem von der rechten:

Kein Filter	Kein Filter
Schwellwertfilter	POMF
Subtraktionsfilter	Frequenzlernen

Wird kein Phase Only Matched Filter und kein Frequenzlernen verwendet, kann die Korrelation durch Kreuzkorrelationen durchgeführt werden (Ka-

```
for(int i = 0; i < N; i++)
{
  if(x[i] != 0)
  {
    int j = max(0, i - MAX_OFFSET);
    int j_end = min(N - 1, i + MAX_OFFSET);
    int offset = j - i;
    for(; j < j_end; j++)
    {
      offsets[offset++] += x[i] * y[j];
    }
  }
}
```

Abbildung 3.4: Vereinfachte Darstellung des verwendeten Algorithmus zur Berechnung aller relevanten Kreuzkorrelationen, x und y stehen für linkes und rechtes Signal, offsets enthält hinterher die Korrelationswerte aller Verschiebungen.

pitel 2.3), andernfalls ist eine Korrelation im Frequenzraum nötig (Kapitel 2.4). Beide liefern dasselbe Ergebnis, die Kreuzkorrelation ist jedoch schneller.

Dies liegt daran, dass mit ihr nicht jede Verschiebung berechnet werden muss, sondern nur bis zu einer maximalen positiven und negativen Verschiebung, die höchstens auftreten kann. Diese ergibt sich aus der Zeit, die ein Geräusch braucht, um die Distanz zwischen den beiden Mikrofonen mit Schallgeschwindigkeit zurück zu legen. Der Wert liegt in dieser Arbeit, abhängig von der Distanz zwischen den Mikrofonen und der Sampling-Rate, etwas unter 40 Samples. Somit reichen Kreuzkorrelationen mit allen Verschiebungen von -40 bis 40 aus, was 81 Verschiebungen ergibt, weitaus weniger als die doppelte Länge einer Folge.

Die Kreuzkorrelation wurde so implementiert, dass sie alle möglichen Verschiebungen auf einmal berechnet. Der Code (siehe Abbildung 3.4) soll den Algorithmus verständlich machen, es handelt sich dabei aber um eine vereinfachte Version und nicht um den originalen Code, der implementiert wurde, da dieser Optimierungen enthält, die das Lesen erschweren.

Wenn das linke Signal x viele Nullen enthält, ist dieser Algorithmus schneller als einer, der die Kreuzkorrelationen für die einzelnen Verschiebungen nacheinander ausrechnet. Ist ein Wert in x Null, werden alle Multiplikationen für die verschiedenen Verschiebungen mit ihm übersprungen. Wurde vorher das Schwellwertfilter oder Subtraktionsfilter auf das Signal angewendet, tritt dieser Fall sehr oft ein. Der Worst Case, in dem alle Werte in x ungleich Null sind, bleibt gleich.

Für die Transformationen zwischen Frequenz und Zeitraum zur Korrelation im Frequenzraum, wie sie das Phase Only Matched Filter und Frequenzen lernende Filter brauchen (Kapitel 2.4), wurde das General Purpose FFT Package[16] benutzt, um das Rad nicht neu erfinden zu müssen. Zwar gilt FFTW[17] hierfür als Quasistandard, läuft jedoch unter GNU General Public License (GPL), was eine Veröffentlichung des eigenen Codes bedeuten würde. Da Doggy zu einem kommerziellen Produkt entwickelt werden soll, hätte diese

Entscheidung später zu Problemen führen können.

Ähnlich wie für den Noise-Level, müssen für das Frequenzen lernende Filter Konstanten bestimmt werden(2.24). Dafür haben sich $L = 10$ und $\alpha = 0.01$ in frühen Tests als brauchbar erweisen.

Wie zu Beginn des Kapitels erwähnt, wird neben der besten Verschiebung auch deren Qualität berechnet. Der Korrelationswert steht bereits dafür, wie gut die Korrelation für eine bestimmte Verschiebung ist, doch dieser Wert kann stark schwanken, abhängig davon, wie laut die Geräusche waren, die zu diesem Zeitpunkt aufgezeichnet wurden. Um einen besseren Vergleich zu haben, wird aus dem Korrelationswert $K_{xy}(r)$ eine Qualität $Q_{xy}(r)$ berechnet, die lautstärkeunabhängig ist und zwischen Null und Eins liegt(3.2).

$$Q_{xy}(r) = \frac{K_{xy}(r)}{\max\{K_{xx}(0), K_{yy}(0)\}} \tag{3.2}$$

Der Korrelationswert $K_{xy}(r)$ wird durch einen weiteren perfekten Korrelationswert dividiert. Perfekt ist ein Korrelationswert immer, wenn ein Signal mit sich selbst korreliert wird. Da es zwei Signale gibt, die unterschiedlich laut sein können, werden beide mit sich selbst mit einer Verschiebung von Null korreliert und der höhere Korrelationswert ist perfekt, kann also hier nicht übertroffen werden.

Es gibt zwei Ausnahmefälle für die Berechnung der Qualität zu einer Verschiebung. Wird das Phase Only Matched Filter benutzt, ist der Korrelationswert bereits unabhängig von der Lautstärke und liegt zwischen Null und Eins, er kann also unmittelbar auch als Qualität verwendet werden. Die zweite Ausnahme wird in dieser Arbeit beim Frequenzen lernenden Filter gemacht. Durch seine Anwendung sinkt der Korrelationswert $K_{xy}(r)$ und somit auch die Qualität $Q_{xy}(r)$. Um die Qualität vergleichbar zu halten, wird der Korrelationswert bei angewendetem Frequenzen lernenden Filter daher um den Faktor zwei erhöht. Dadurch kann auch eine Qualität höher als Eins erreicht werden. Alternativ hätte man auch die Konstanten im Model anpassen können.

3.4 Model

Der ROS-Knoten Perception publiziert mehrach pro Sekunde ein SOI Percept (Kapitel 3.3). In diesem steht die am besten passende Verschiebung der Signale vom linken und rechten Mikrofon sowie die Qualität dieser Verschiebung. Das Model wird durch jedes SOI Percept aktiv und berechnet aus der Verschiebung die Richtung zur Geräuschquelle. Über einen bestimmten Zeitraum bilden mehrere SOI Percepte kombinierte Qualitäten für alle Richtungen. Ist die Qualität für eine Richtung hoch genug, wird sie als Richtung zu einer interessanten Geräuschquelle in ein Modelled SOI Percept (Kapitel 3.2) geschrieben und so in das laufende ROS publiziert.

Jedes SOI Percept wird in einem Ringpuffer der Größe zehn gespeichert, so sind in ihm immer die letzten zehn SOI Percepte verfügbar. Die Größe Zehn wurde gewählt, da diese der Formel(3.1) nach knapp einer halben Sekunde entspricht. Zwei SOI Percepte, zwischen denen eine halbe Sekunde oder mehr Abstand liegt, können also nicht gemeinsam zu der Hypothese eines interessanten Geräusches beitragen.

Bei jedem neuen SOI Percept wird geprüft, ob es zu einem neuen interessanten Geräusch beitragen kann. Dazu wird als erstes seine Qualität gegen einen Schwellwert von 0.08 getestet. Leere SOI Percepte von Folgen, in denen der Noise-Level nicht überschritten wurde, haben eine Qualität von 0 und liegen somit automatisch unter dem Schwellwert. Wird der Schwellwert nicht erreicht, fließt das SOI Percept in keine der folgenden Rechnungen mit ein.

Anschließend wird aus der Verschiebung im SOI Percept die Richtung errechnet, aus der das dazugehörige Geräusch kam (Kapitel 3.3). Liegt dieses Geräusch außerhalb eines bestimmten Öffnungswinkels unter 180°, soll es nicht interessant sein (Kapitel 2.1) und fließt ebenfalls nicht in die folgenden Rechnungen mit ein.

Um aus mehreren SOI Percepten die Richtung zu einem interessanten Geräusch zu berechnen, wird unter ihnen die weiteste Verschiebung ins Negative sowie die weiteste Verschiebung ins Positive heraus gesucht. Für diese Verschiebungen r und alle, die dazwischen liegen, wird eine kombinierte Qualität $Q(r)$aus allen einfließenden SOI Percepten ermittelt(3.3).

$$Q(r) = \sum_{m=0}^{M-1} q_m e^{-\frac{1}{s}\ (r_m - r)^2} \tag{3.3}$$

Die Variable m ist der Index eines SOI Percepts im Ringpuffer und M die Größe des Ringpuffers. Die Summe läuft also für jede Verschiebung r über alle SOI Percepte. Für die Standardabweichung s der hier verwendeten Normalverteilung hat sich ein Wert von 6 in frühen Tests als brauchbar erwiesen. Die Verschiebung eines SOI Percepts ist r_m und ihre Qualität q_m.

Ziel der Berechnungen(3.3) ist es das höchste $Q(r)$ zu finden. Liegt dieses oberhalb eines weiteren Schwellwerts, für den sich 0.14 als brauchbar erwiesen hat, wird ein interessantes Geräusch bei der Verschiebung r festgestellt.

Die Berechnung(3.3) wird für diskrete Verschiebungen r durchgeführt, diese müssen aber nicht ganzzahlig sein. Perception kann nur SOI Percepte mit Verschiebungen um ganze Samples liefern, aber kombiniert kann ein höchstes $Q(r)$ auch an einem r mit Nachkommastellen liegen. Aus diesem Grund liegt zwischen den diskreten Verschiebungen r, mit denen die Berechnung(3.3) durchgeführt wird, kein Abstand von 1 sondern 0.1. Kleinere Schritte würden exaktere Ergebnisse liefern, vergrößern aber auch die Anzahl benötigter Rechenschritte.

Die Berechnungen werden nur ausgeführt, wenn das neu hinzugekommene SOI Percept auch in diese einfließen soll. Andernfalls könnte ein SOI Percept ohne eigentlichen Einfluss in die Rechnung ein neues interessantes Geräusch bewirken, dadurch dass es ein älteres aus dem Ringpuffer verdrängt hat.

Wenn ein interessantes Geräusch festgestellt wurde, wird aus seiner Verschiebung r die zugehörige Richtung errechnet(2.3). Ist der Unterschied dieser Richtung zu der des letzten interessanten Geräusches größer als 6°, wird vom Model ein Modelled SOI Percept ins ROS publiziert, in dem diese Richtung enthalten ist.

Letztlich soll sich der Roboter in die Richtung eines interessanten Geräusches drehen, doch möglicherweise wirkt Doggys Verhalten menschlicher, wenn er sich stärker geradeaus konzentriert. Dazu wurden im Rahmen dieser Arbeit zwei Parameter implementiert, die den ROS-Knoten Model und sein Ergebnis beeinflussen. Der erste Parameter erhöt die Qualität eines SOI Percepts, je kleiner der Betrag der Verschiebung ist, also je mittiger das dazugehörige Geräusch vor dem Roboter liegt. Der zweite Parameter wird mit dem α multipliziert, das als

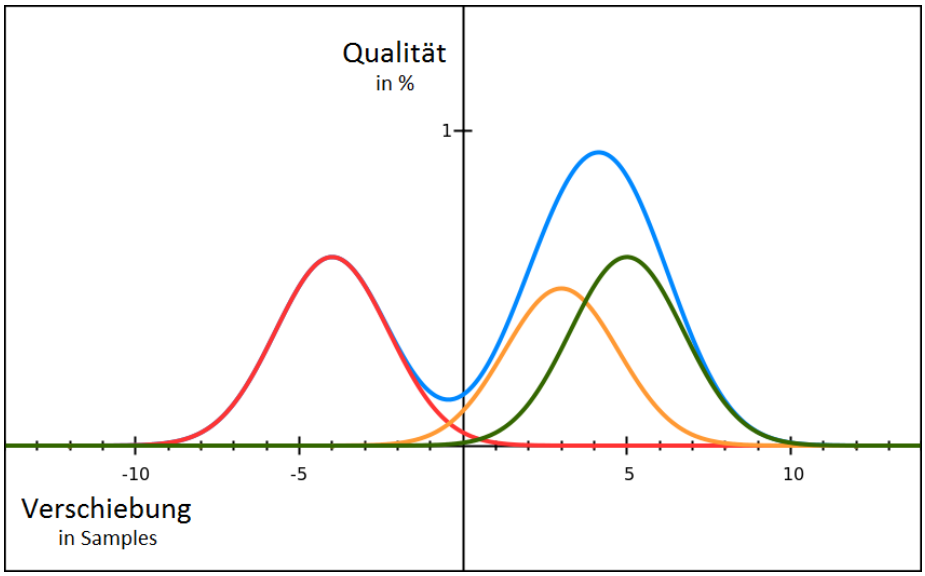

Abbildung 3.5: Qualitäten der Verschiebungen: Rot, gelb und grün sind SOI Percepte an ihrer Verschiebung (-4, 3 und 5), multipliziert mit Normalverteilung der Standardabweichung 6 und ihrer Qualität (0.6, 0.5 und 0.6), blau ist die Summe aller Graphen entsprechend $Q(r)$ aus Formel(3.3).

Richtung des interessanten Geräusches errechnet wurde(2.3). Dadurch kann eine Drehung abgeschwächt werden, je weiter diese Doggy zur Seite drehen würde.

3.5 Motion

Wenn Model die Richtung eines interessanten Geräusches errechnet hat, publiziert es diese Richtung in einem Modelled SOI Percept in das laufende ROS. Der ROS-Knoten Motion wird dadurch aktiv und liest diese Richtung aus, um eine Bewegung zu berechnen, die den Roboter zur Geräuschquelle dreht. Diese Bewegung wird als Joints Request publiziert.

Für die Bewegung braucht nur einer der drei Motoren des Roboters einen bestimmten Winkel anzusteuern. Dieser ist bereits der Winkel des Modelled SOI Percepts mit verändertem Vorzeichen. Das falsche Vorzeichen kommt dadurch zustande, dass es bei der Verschiebung Standard ist das zweite, also rechte Signal, im Verhältnis zum Ersten linken zu verschieben. Dadurch erzeugt ein Geräusch, welches das linke Mikrofon früher erreicht, also von links kommt, eine negative Verschiebung, woraus auch ein negativer Winkel resultiert. Da Winkel von Motoren gegen den Uhrzeigersinn positiv drehen, würde dieser Winkel den Roboter nach rechts drehen. Daher muss das Vorzeichen umgedreht werden.

Mehr Aufgaben erfüllt Motion nicht, so wie es im Rahmen dieser Arbeit implementiert wurde. Dennoch hat es die Berechtigung ein eigener ROS-Knoten zu sein, da es aus der Kenntnis über ein interessantes Geräusch eine Absicht macht. Auch wenn Doggy diese Entscheidung nach derzeitigem Stand sehr einfach fällt. Ein zukünftiges komplexeres Verhalten würde an dieser Stelle in den Prozess eingreifen.

3.6 Control

Der ROS-Knoten Control ist die Schnittstelle zwischen ROS und Doggys Mikrocontroller. Wird im laufenden ROS ein Joints Request publiziert, beispielsweise von Motion, liest Control dies und wandelt es zu einer Nachricht um, die es an den Mikrocontroller sendet. Zusätzlich empfängt Control permanent Nachrichten des Mikrocontrollers, die Informationen über aktuelle Zustände der Motoren enthalten. Diese verpackt Control in eine ROS-Message namens Joints (Kapitel 3.2) und publiziert sie seinerseits ins ROS.

Die Nachrichten, die zwischen Control und Mikrocontroller ausgetauscht werden, sind Zeichenketten mit maximaler Länge von 64Byte. Um diese Nachrichten über die USB-Schnittstelle zu übertragen, wird die C++ Bibliothek LibSerial[18] verwendet.

Nachrichten an den Mikrocontroller starten mit dem Schlüsselwort „ang", gefolgt von den Zielwinkeln für die drei Motoren als Fließkommazahlen. Als Trennung dient ein Leerzeichen.

Nachrichten vom Mikrocontroller beziehen sich immer nur auf einen der drei Motoren, Control fasst aber alle drei Motoren in Joints zusammen. Zusätzlich zum aktuellen Winkel und Zielwinkel, schickt der Mikrocontroller auch Informationen über die Belastung der Motoren. Bisher werden Nachrichten vom Typ Joints im ROS von keinem Knoten gelesen.

Kapitel 4

Tests

In dem vorherigen Kapitel wurde beschrieben, wie Doggy mit einem System ausgestattet wurde, dass es ihm ermöglicht sich zu interessanten Geräuschen zu drehen. Dabei wurden mehrere Filter implementiert, die in neun unterschiedlichen Kombinationen benutzt werden können. In diesem Kapitel wird zunächst ein Test ohne Probanden beschrieben, in dem diese Kombinationsmöglichkeiten ausführlich auf ihre Zuverlässigkeit und Genauigkeit unter verschiedenen Bedingungen geprüft werden. Anschließend wird auf einen zweiten Test mit Probanden eingegangen, der mit nur einer bestimmten Filter-Kombination durchgeführt wurde, die sich in dem ersten Test als Beste erwiesen hat. Dabei wurde Doggy unter realistischen Bedingungen vor eine Gruppe von Studenten gestellt, die ihn nicht kannten und auf deren Geräusche er reagieren sollte.

4.1 Ohne Probanden

Ziel dieses Tests war es, die neun unterschiedlichen Filter-Kombinationen auf ihre Zuverlässigkeit und Genauigkeit zu prüfen. Dazu fand der Test unter möglichst kontrollierten Bedingungen statt, unter denen der Roboter Sound-Dateien aufzeichnete. Es wurden 588 Geräusche an verschiedenen Positionen zum Roboter aufgezeichnet. Anschließend wurden die Filter-Kombinationen auf diesen Sound-Dateien getestet und ihre Ergebnisse in Tabellen gesammelt. Die neun Filter-Kombinationen sind:

- Keine Filter (no-no)
- Schwellwertfilter (thr-no)
- Subtraktionsfilter (sub-no)
- POMF (no-pomf)
- Schwellwertfilter mit POMF (thr-pomf)
- Subtraktionsfilter mit POMF (sub-pomf)
- Frequenzlernen (no-learn)
- Schwellwertfilter mit Frequenzlernen (thr-learn)
- Subtraktionsfilter mit Frequenzlernen (sub-learn)

Alle Filter sind beschrieben unter Signalfilter (Kapitel 2.5).

4.1.1 Testaufbau

Für den Test wurden vier bestimmte Geräusche verwendet: Klatschen, lautes Räuspern, Auftrumpfen eines kleinen Balls und „123" rufen. Diese Geräusche wurden jedes Mal manuell und möglichst gleich erzeugt.

Die Geräusche sollten in verschiedenen Entfernungen und an verschiedenen Winkeln zum Roboter erzeugt werden. Dazu wurden einem bestimmten Muster folgend (siehe Abbildung 4.1) in jedem Testgelände Markierungen für 49 Geräuschpositionen auf den Boden geklebt.

Zusätzlich zu unterschiedlichen Geräuschen und unterschiedlichen Positionen zum Roboter, sollten auch unterschiedliche Umgebungen Teil des Tests sein. Darum wurden drei Testumgebungen ausgesucht. Die Erste war im Raum MZH1400 der Universität Bremen, ein großer Raum ohne besondere Merkmale, in dem alle Tische und Stühle am Rand gestapelt wurden. Dort gab es Echos, aber keine sonstigen Störgeräusche. Als zweite Testumgebung diente wieder der Raum MZH1400, dieses Mal wurde allerdings als künstliche Störquelle von der Position 50° 3m permanent ein Lied mit singenden Kindergartenkindern[19] abgespielt. Die dritte Testumgebung lag draußen direkt neben einer Baustelle (siehe Abbildung 4.2), dort gab es permanente Störgeräusche von Baustellenfahrzeugen und starkem Wind, der auch durch Doggys Kostüm wehte.

Doggy trug bei den Tests sein Kostüm und sein Mikrocontroller war eingeschaltet, sodass seine Motoren ständige Vibrationen und Störgeräusche erzeugten. In der dritten Testumgebung an der Baustelle wurde vergessen den Mikrocontroller einzuschalten, die Störgeräusche wären aber vermutlich unerheblich gewesen gegen den Baustellenlärm und den Wind.

ROS lief nicht während die Sound-Dateien aufgenommen wurden, sodass Doggy während des Tests nicht auf interessante Geräusche reagierte.

Die Winkel zu den Geräuschen an den Geräuschpositionen stimmten nicht exakt mit den Winkeln des Musters (siehe Abbildung 4.1) überein. Dies lag zum einen daran, dass die Geräusche nicht auf Höhe der Mikrofone erzeugt wurden. Darum wurden im Nachhinein die Höhen der vier verschiedenen Geräusche festgestellt und für jedes die Winkel aller Geräuschpositionen neu berechnet.

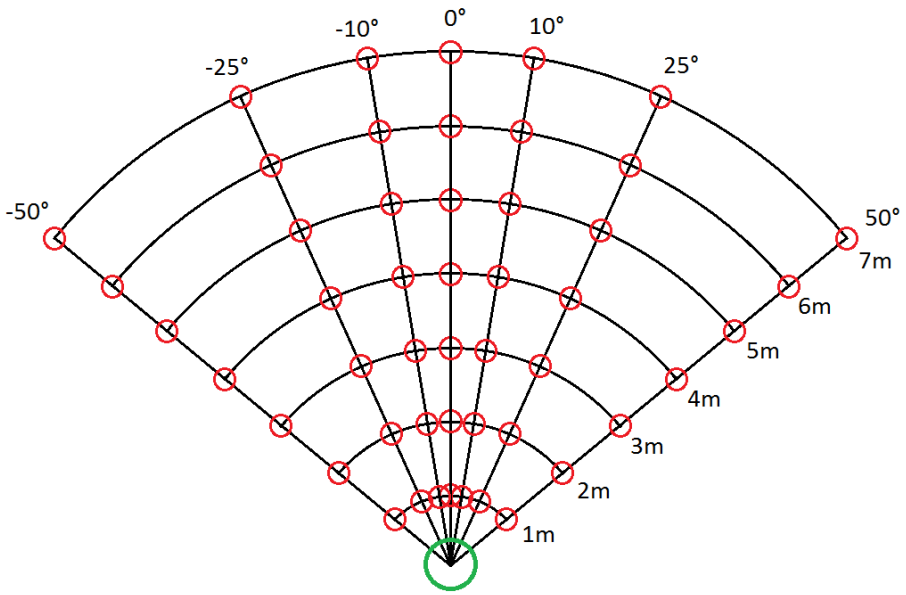

Abbildung 4.1: Skizze der Geräuschpositionen, grüner Kreis: Roboter, rote Kreise: Geräuschpositionen.

Abbildung 4.2: Testgelände für den Versuch vor der Baustelle aus Sicht des Roboters, Markierungen auf dem Boden für Geräuschpositionen.

Zum anderen stand Doggy in MZH1400 unbeabsichtigt um 2° gedreht, was ebenfalls für die Geräuschpositionen eingerechnet wurde. Des Weiteren entstand ein Fehler unbekannter Größe dadurch, dass die Testgeräusche manchmal möglicherweise wenige Zentimeter neben den Bodenmarkierungen gemacht wurden, was sich bei kurzen Distanzen stärker auswirkt als auf hohen.

4.1.2 Ergebnisse

Die Ergebnisse stammen aus 12 Soundaufnahmen mit insgesamt 588 Geräuschen, die alle ungeschnitten auf dem beiliegenden Datenträger zu finden sind. Auf ihnen wurden neun unterschiedliche Kombinationen von Filten getestet. Die detaillierten Ergebnisse befinden sich in einer Excel-Datei, die ebenfalls auf dem dieser Arbeit beiliegenden Datenträger ist.

Bei dem Test wurden die Winkel des angepassten Musters der kontrolliert erzeugten Geräusche verglichen mit den Winkeln, zu denen Doggy sich gedreht hätte. Neben der Anzahl korrekt erkannter Geräusche ist dabei auch die Anzahl zusätzlich erkannter Geräusche aufgeführt, die während des Versuches unbeabsichtigt erzeugt wurden oder Störgeräusche sind, die als interessant klassifiziert wurden. Als Ausreißer wurden erkannte Geräusche gezählt, deren Winkel um mehr als 10° abweicht, dennoch zählen sie auch als erkannt. Der Fehler ist die Differenz zwischen den Winkeln.

Die Ergebnisse vom Klatschen im MZH1400 (siehe Tabelle 4.1) und vor der Baustelle (siehe Tabelle 4.2) sind besonders interessant, da sie die Situationen mit besonders wenigen und besonders vielen Störgeräuschen abbilden. Eine Zusammenfassung aller Ergebnisse wurde ebenso erstellt (siehe Tabellen 4.3).

Tabelle 4.1: Ergebnis von 49 Geräuschen (Klatschen), Angaben in Grad, Raum MZH1400 ohne Kinderlied.

	no-no	thr-no	sub-no	no-pomf	thr-pomf	sub-pomf	no-learn	thr-learn	sub-learn
Erkannt	49	48	49	48	47	49	49	48	49
Ausreißer	0	0	0	0	0	0	0	0	0
Zusätzlich	1	1	1	3	0	1	1	1	1
Fehler Mittel	-0.55	-0.51	-0.47	0.04	-0.28	-0.05	-0.27	-0.45	-0.23
Fehler Min	-4.17	-4.17	-4.17	-4.25	-7.92	-2.66	-4.17	-4.17	-4.17
Fehler Max	2.39	2.64	2.64	3.16	2.64	3.16	3.16	2.64	3.16
Fehler StdAbw.	1.36	1.41	1.43	1.28	1.72	1.22	1.41	1.37	1.49

Tabelle 4.2: Ergebnis von 49 Geräuschen (Klatschen), Angaben in Grad, Baustelle.

	no-no	thr-no	sub-no	no-pomf	thr-pomf	sub-pomf	no-learn	thr-learn	sub-learn
Erkannt	39	36	37	40	40	36	39	35	38
Ausreißer	0	0	0	3	2	1	0	0	0
Zusätzlich	8	3	3	9	6	3	0	0	1
Fehler Mittel	0.82	0.26	0.15	0.93	2.19	1.29	1.02	-0.38	0.74
Fehler Min	-3.97	-15.22	-18.71	-29.88	-14.36	-4.16	-2.02	-13.98	-2.60
Fehler Max	6.02	6.02	6.02	36.23	33.63	16.68	6.02	6.02	6.02
Fehler StdAbw.	1.89	3.46	3.72	8.81	6.53	3.62	1.64	4.68	1.78

Tabelle 4.3: Ergebnisse von jeweils 196 Geräuschen (49 Klatschen, 49 Räuspern, 49 Ball Auftrumpfen und 49 „123" rufen), Angaben in Grad, oben: Raum MZH1400 ohne Kinderlied, mitte: Raum MZH1400 mit Kinderlied, unten: Baustelle.

	no-no	thr-no	sub-no	no-pomf	thr-pomf	sub-pomf	no-learn	thr-learn	sub-learn
Erkannt	195	193	185	191	182	191	193	177	174
Ausreißer	0	1	0	4	4	1	0	1	0
Zusätzlich	7	6	1	30	17	13	5	12	3
Fehler Mittel	-1.07	-1.10	-1.03	0.66	-0.52	-0.66	-0.45	-0.85	-0.59
Fehler Min	-10.67	-10.67	-7.98	-47.94	-42.26	-18.03	-7.51	-14.74	-7.51
Fehler Max	8.75	25.66	8.75	51.94	51.94	18.26	9.43	11.49	6.49
Fehler StdAbw.	2.25	3.19	2.17	8.19	7.04	3.19	2.01	2.82	2.02

	no-no	thr-no	sub-no	no-pomf	thr-pomf	sub-pomf	no-learn	thr-learn	sub-learn
Erkannt	195	185	181	182	168	181	192	170	164
Ausreißer	0	0	0	2	3	0	0	0	0
Zusätzlich	2	5	2	23	17	5	2	5	2
Fehler Mittel	-1.05	-1.21	-0.95	-0.31	-0.39	-0.37	-0.37	-1.06	-0.80
Fehler Min	-24.61	-36.43	-36.43	-46.93	-70.28	-13.33	-5.31	-36.43	-36.43
Fehler Max	6.49	6.49	6.49	45.80	15.71	8.70	6.49	9.57	6.49
Fehler StdAbw.	2.89	3.95	3.32	7.17	6.70	2.49	1.95	4.43	3.84

	no-no	thr-no	sub-no	no-pomf	thr-pomf	sub-pomf	no-learn	thr-learn	sub-learn
Erkannt	146	128	118	148	130	133	140	117	104
Ausreißer	0	0	1	8	6	4	0	1	0
Zusätzlich	11	8	4	26	20	16	1	5	6
Fehler Mittel	0.13	0.10	0.44	1.23	2.02	1.16	0.48	0.07	0.82
Fehler Min	-9.26	-15.22	-18.71	-29.88	-21.71	-6.45	-4.28	-13.98	-5.04
Fehler Max	9.02	8.87	17.76	46.85	49.19	42.77	9.02	12.34	8.23
Fehler StdAbw.	2.54	2.98	3.48	8.30	9.11	5.17	2.02	3.52	2.33

Tabelle 4.4: Rechenzeiten der Filter-Kombinationen in Millisekunden, (cc) steht für die Berechnung durch Kreuzkorrelationen (Kapitel 2.3) ohne Korrelation im Frequenzraum (Kapitel 2.4).

Rechenzeit	Min	Max
no-no (cc)	0	3
thr-no (cc)	0	1
sub-no (cc)	0	1
no-no	1	3
thr-no	1	3
sub-no	1	3
no-pomf	1	5
thr-pomf	1	5
sub-pomf	1	5
no-learn	1	5
thr-learn	1	5
sub-learn	1	5

Die von den Filter benötigte Rechenzeit konnte nur mit einer Genauigkeit von einer Millisekunde erfasst werden und schwankte abhängig von der Menge an Störgeräuschen (siehe Tabelle 4.4).

4.1.3 Auswertung

Der Test ohne Probanden hat gezeigt, dass die meisten Filter-Kombinationen nur geringe Fehler bei der Berechnung der Winkel machen. Die Standardabweichung dafür liegt fast immer unter 4°. Auffällig dabei sind die Ergebnisse mit dem Phase Only Matched Filter (Kapitel 2.5.3). Sie haben im Durchschnitt den höchsten Fehler, jedoch bei sehr wenigen Störgeräuschen den niedrigsten (siehe Tabelle 4.1). POMF hat zudem mit Abstand die meisten Ausreißer und zusätzlich erkannten Geräusche. Dies war zu erwarten, denn POMF kann zwar die Verschiebung zweier Signale stark verdeutlichen, entfernt dabei aber gleichzeitig alle Informationen über die Lautstärke der Geräusche. Hört Doggy mehrere Geräusche, entscheidet er sich für eines davon, jedoch nicht unbedingt für das lautere. In Situationen, in denen alle Störgeräusche leise sind, scheint das Subtraktionsfilter (Kapitel 2.5.2) dem POMF zu helfen, da sie durch dieses entfernt werden. Dennoch gehören die Ergebnisse in den meisten Fällen zu den schlechtesten. Durch ein Schwellwertfilter (Kapitel 2.5.1) verbessern sich die Ergebnisse manchmal, werden zum Teil aber auch schlechter.

Ohne ein im Frequenzraum arbeitendes Filter sind die Ergebnisse bereits relativ gut, mit einer niedrigen Standardabweichung des Fehlers. Auch Ausreißer sind selten, dafür werden aber noch relativ viele zusätzliche Geräusche erkannt, beispielsweise ohne Einsatz eines Filters: 11 bei 196 Testgeräuschen an der Baustelle (siehe Tabelle 4.3). Verbessert werden können diese Ergebnisse in den Tests nur durch ein Subtraktionsfilter und auch nur unter Bedingungen mit wenig Störgeräuschen im MZH1400 ohne Kinderlied. Diese Filterkombinationen sind nicht auf den Frequenzraum angewiesen und können mit der geringsten Rechenzeit berechnet werden (siehe Tabelle 4.4).

Das beste Ergebnis erzielt das Frequenzen lernende Filter (Kapitel 2.5.4).

Über alle vier Geräusche betrachtet hat dieses in jeder Umgebung die niedrigste Standardabweichung des Fehlers, wenn es mit keinem weiteren Filter kombiniert wird. Die Anzahl korrekt erkannter Geräusche liegt im Vergleich im Mittelfeld, dafür ist die Anzahl an Ausreißern und zusätzlich erkannten Geräuschen sehr niedrig. Besonders auffällig ist dies beim Klatschen als Geräusch an der Baustelle. Dabei werden ohne Filter 39 von 49 Geräuschen erkannt und 8 zusätzliche. Mit Frequenzen lernendem Filter werden ebenfalls 39 erkannt, jedoch kein einziges zusätzliches. Offensichtlich funktioniert es den Einfluss von Störgeräuschen damit zu reduzieren, wenn diese auf anderen Frequenzen liegen, als die interessanten Geräusche.

Alle Filterkombinationen brauchen auf einem Computer wie dem hier verwendeten kaum Rechenzeit.

Als beste Filterkombination für Doggys Anwendungsbereich gilt nach Aussage dieser Auswertung das Frequenzen lernende Filter ohne Kombination mit einem weiteren Filter.

4.2 Mit Probanden

Doggys neue Fähigkeit zum stereo Hören soll praxistauglich sein, daher war es notwendig sie auch mit echtem Publikum zu testen. Bei diesem Versuch wurden keine Soundaufnahmen gemacht um später die Algorithmen darauf zu testen, sondern Doggy hat unmittelbar auf die Geräusche der Probanden reagiert. Der Test wurde zweifach ausgeführt und um die Reaktionen bewerten zu können dabei aus unterschiedlichen Positionen gefilmt.

4.2.1 Testaufbau

Für diesen Test wurde eine bestimmte Kombination von Filtern ausgewählt, die sich im vorherigen Test ohne Probanden als Erfolgversprechendste erwiesen hat. Die Wahl fiel auf das Frequenzen lernende Filter (Kapitel 2.5.4) ohne vorheriges Schwellwertfilter (Kapitel 2.5.1) oder Subtraktionsfilter (Kapitel 2.5.2). Die beiden Parameter, die Doggy sich stärker nach vorne konzentrieren lassen (Kapitel 3.4), wurden nicht verwendet.

Der Test fand an der Uni Bremen im Cartesium Raum C001 statt (siehe Abbildung 4.3). Doggy wurde in dem Raum vor eine Gruppe von Studenten gestellt, die ihn noch nicht kannten (siehe Abbildung 4.4).

Der Test wurde zweifach durchgeführt mit unterschiedlichen Studenten. Beim ersten Mal war Doggy ohne Oberteil und ohne Kostüm. An seiner Hüfte war eine kleine Kamera befestigt, die nach vorne gerichtet war und sich mitdrehte. Dabei wurde also immer in Blickrichtung des Roboters gefilmt. Beim zweiten Durchlauf war Doggy mit seinem Oberteil und dem Kostüm ausgestattet, die Kamera filmte dabei aus einer Ecke des Raumes seine Reaktionen und die Probanden.

Den Probanden wurde zu Anfang nichts über die Aufgabe oder Funktionsweise von Doggy gesagt, stattdessen bekamen sie die Anweisung die Aufmerksamkeit des Roboters durch Geräusche auf sich zu lenken. Nach einer Weile kam ein roter Ball hinzu. Die Studenten wurden aufgefordert ein auffälliges Geräusch zu machen, wenn sie den Ball haben und ihn anschließend zu jemand anderem weiter zu werfen. Beim zweiten Durchlauf, bei dem aus einer Ecke gefilmt wurde, wurden die Probanden später zusätzlich aufgefordert durch Gespräche eine

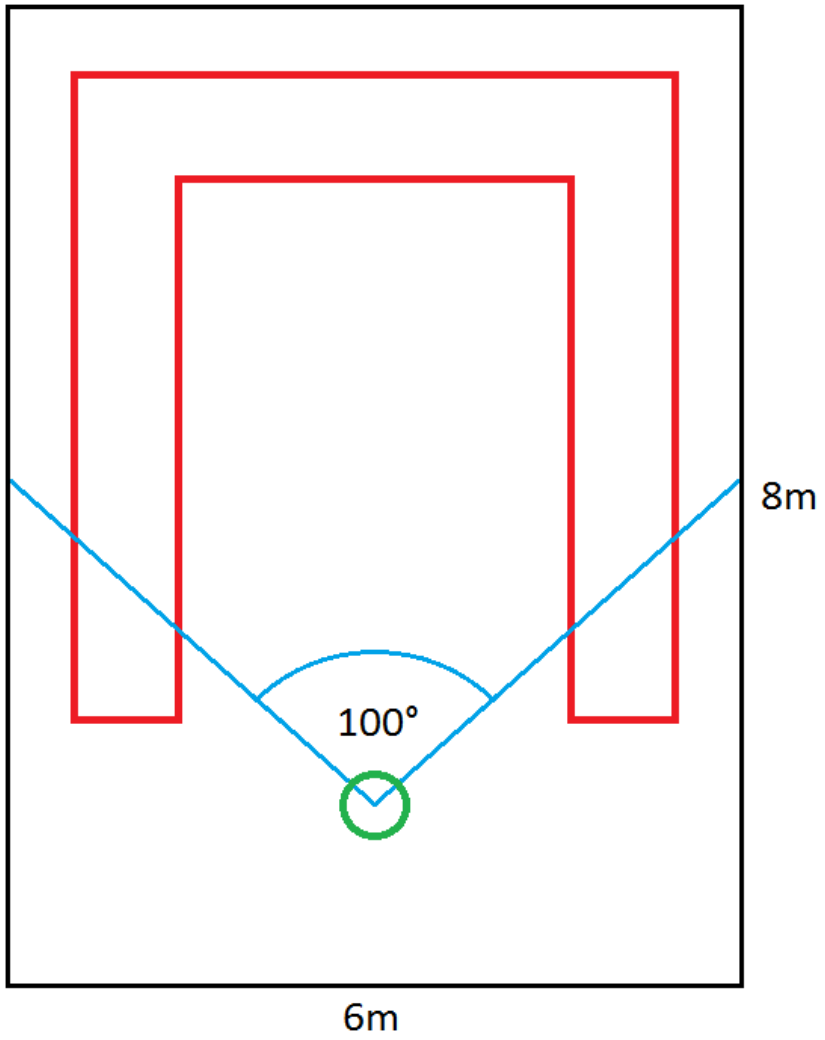

Abbildung 4.3: Skizze der Testumgebung im Raum C001, grüner Kreis: Doggy, blau: Öffnungswinkel, rot: Tische mit Probanden.

Abbildung 4.4: Doggy wurde für den Test vor seine Probanden gestellt.

Geräuschkulisse zu erzeugen.

4.2.2 Ergebnisse

Als Ergebnis des Tests mit Probanden sind zwei kurze Videos entstanden, die beide ungeschnitten auf dem mitgelieferten Datenträger zu finden sind.

Das erste Video filmt aus der Sicht des Roboters. Man kann darin sehen, dass Doggy auf die meisten Geräusche korrekt reagiert und den betreffenden Probanden anguckt. Beispielsweise reagiert er im Video an der Stelle 1:50 min präzise auf zwei Probanden, die am anderen Ende des Raumes in ein Taschentuch niesen. An den Reaktionen des Roboters kann man sehen, dass er sich der Lautstärke im Raum gut anpasst. So hört er für wenige Sekunden auf sich zu drehen, wenn alle im Raum versuchen die Aufmerksamkeit zu bekommen oder anfangen laut zu lachen.

Manche Probanden versuchen zu Anfang die Aufmerksamkeit durch leise Geräusche auf sich zu lenken, was meistens nicht funktioniert, gerade wenn derjenige weit entfernt sitzt.

Doggy dreht sich nach einer weiten Drehung nach links anschließend oft wieder ein Stück zurück nach rechts, weg von dem Geräusch. Zusätzlich dreht er sich einmal als Reaktion auf ein interessantes Geräusch um mehr als 10° von dem verursachenden Probanden weg bei 3:16 min. Gegen Ende des Videos bei 3:50 min ruft ein Proband „Wuhahaha!", später bei 3:56 min ein anderer „Booom!", auf beide reagiert Doggy leider nicht.

Das zweite Video filmt aus einer Ecke des Raumes, darin reagiert Doggy ähnlich gut, wie im ersten. Mehrfach pfeifen Probanden ohne dabei sehr laut zu sein und Doggy reagiert korrekt. Lacht ein einzelner Proband, dreht sich Doggy zu ihm, lachen viele aus verschiedenen Richtungen wie bei 7:04 min, reagiert er nicht. Am besten reagiert er immer wieder auf einen Probanden ganz vorne rechts, der vermutlich gerade noch innerhalb des Öffnungswinkels sitzt und am nächsten am Roboter. Bei 5:12 min und 8:28 min betritt ein verspäteter Student den Raum und die Tür macht ein lautes Geräusch, auf das Doggy aber korrekterweise nicht reagiert, da es außerhalb des Öffnungswinkels liegt.

Ab 6:10 min erzeugen die Probanden durch Gespräche eine laute Geräuschkulisse, dennoch reagiert Doggy unverändert auf interessante Geräusche, die sich in ihrer Lautstärke davon absetzen.

Es ist nicht zu merken, dass Doggy durch das Tragen des Kostüms im zweiten Video schlechter reagiert, als im ersten. Im zweiten Video mit Kostüm tritt das Nachdrehen nach einer weiten Drehung nach links nicht mehr auf.

Es kam in beiden Videos nie vor, dass Doggy reagiert hat ohne ein auffälliges Geräusch gehört zu haben. Es passierte nur ein einziges Mal, dass er sich in eine falsche Richtung gedreht hat, das erwähnte Nachdrehen im ersten Video nicht mitgezählt.

Es bleibt festzuhalten, dass die Probanden im Test mit Doggy offensichtlich Spaß hatten und das Interesse der Studenten an dem Roboter geweckt wurde.

4.2.3 Auswertung

Doggy erkennt die meisten auffällige Geräusche als interessant und dreht sich ihnen korrekt zu ohne sich dabei von Störgeräuschen beeinflussen zu lassen. Es zeigen sich aber auch Schwächen, so spielt die Lautstärke eines Geräusches scheinbar die wichtigste Rolle. Einige durchaus interessante Geräusche der Probanden wurden von ihm nicht als solche wahrgenommen, da sie sich in ihrer

Lautstärke nicht ausreichend von der Umgebung abgehoben haben. Dabei spielt aber auch die Entfernung zum Roboter eine wichtige Rolle, denn als der Platz nahe rechts am Roboter im zweiten Video besetzt war, hat er auf die Person dort sehr gut reagiert.

Auf das Rufen von „Wuhahaha!" und „Booom!" im ersten Video hat Doggy vermutlich nicht reagiert, da in den Worten keine Silbe hart und laut ausgesprochen wurde. Schon bei Tests in frühen Entwicklungsphasen hatte sich gezeigt, dass Doggy auf Wörter, die beispielsweise mit einem hart ausgesprochenen T beginnen wie „Test!", besser reagiert, als auf Wörter mit gleichmäßigem weichen Klang.

Das Einschränken des Öffnungswinkels funktioniert offensichtlich, da Doggy im zweiten Video nicht auf das zweimalige Zufallen der Tür reagiert.

Das Nachdrehen von Doggy im ersten Video, das häufig auftritt, nachdem er sich weit nach links gedreht hat, tritt im zweiten Video nicht mehr auf. Die Ursache für das Nachdrehen muss darin liegen, dass Doggy in dem Moment, in dem er die Drehung vollendet, ein weiteres interessantes Geräusch hört, das weiter in der Mitte vor ihm liegt. Der Verdacht liegt nahe, dass dieses Geräusch beim plötzlichen Abbremsen von Doggys Hüfte ausgelöst wird. Trägt Doggy sein Oberteil und Kostüm, wie im zweiten Video, bremst ihn das in dieser Bewegung scheinbar aus und lässt ihn leiser anhalten, was das Problem löst.

Ein besonders positives Ergebnis des Tests mit Probanden ist, dass es praktisch keine False-Positives gibt. Es lässt sich in dieser Auswertung kein Fall feststellen, in dem Doggy sich dreht, ohne aus der entsprechenden Richtung ein Geräusch gehört zu haben und nur einmal dreht er sich merklich neben eine Geräuschquelle bei 3:16 min im ersten Video.

Kapitel 5

Fazit

Ziel dieser Arbeit war es ein stereo hörendes System auf dem Ballspielroboter Doggy zu implementieren, das ihn zu interessanten Geräuschen dreht. Dieses Ziel wurde erreicht. Dabei liegt die Präzision bei nur rund 2° Standardabweichung. Es konnte auch gezeigt werden, dass dieses System in Doggys typischen Einsatzszenarien mit starken Störgeräuschen funktioniert.

Damit Doggy auf ein Geräusch reagiert, muss es seine Mikrofone laut genug erreichen. Auf eine Entfernung von beispielsweise sieben Metern muss sich ein Geräusch mit seiner Lautstärke schon deutlich von der Umgebung absetzen. Bei Sprache werden Wörter schlechter erkannt, je weicher sie ausgesprochen werden.

Im Rahmen dieser Arbeit wurde das Phase Only Matched Filter getestet. Dabei zeigte sich, dass es eine potenziell höhere Präzision bietet, hier aber ungeeignet ist.

Es wurde ein unübliches Subtraktionsfilter getestet. Mit diesem konnte das Ergebnis des Phase Only Matched Filters in dieser Arbeit verbessert werden.

Im Rahmen dieser Arbeit entstand ein neues adaptives Filter, welches die Häufigkeit von einzelnen Frequenzen lernt. Anhand des Gelernten kann das Frequenzen lernende Filter häufige Frequenzen abschwächen oder sogar ganz ausblenden. Dies hat sich als sehr wirkungsvoll gegen monotone Störgeräusche erwiesen und lieferte insgesamt die besten Ergebnisse.

Kapitel 6

Ausblick

In dieser Arbeit wurde die Implementierung auf Doggys Mikrocontroller nicht verändert. Nach dem derzeitigen Stand kann dieser nur entweder Doggys Oberteil bewegen, um einen geworfenen Ball abzufangen oder Gelenkwinkel vom Computer annehmen, um sich beispielsweise zu einem interessanten Geräusch zu drehen. Was bisher fehlt ist ein Verhalten, welches situationsabhängig entscheiden kann, welche Bewegungen ausgeführt werden können. Dieses könnte in einem ROS-Knoten wie Motion implementiert werden, wenn dieser mehr Informationen darüber hätte, was Doggys Kameras gerade erkennen. Diese Informationen könnten wiederum von einer ROS-Implementierung der Ball- und Flugbahnerkennung stammen, die diese Aufgabe dann dem Mikrocontroller abnimmt. Eine solche Implementierung existiert bereits von Doggys Vorgänger Piggy und soll in Zukunft wieder verwendet werden.

Auch wenn die in dieser Arbeit erreichte Präzision bereits sehr gut ist, gibt es noch Verbesserungspotenzial. Vergrößert man die Entfernung zwischen den beiden Mikrofonen, wird die zeitliche Verschiebung ihrer Signale größer, wodurch die Auflösung der Richtung zunimmt. Gleiches kann theoretisch erreicht werden, indem man die Abtastfrequenz erhöht, also die Anzahl der Samples pro Sekunde. In dieser Arbeit wurde eine Abtastfrequenz von 44100 Hz verwendet, die Soundkarte des verwendeten Computers ermöglicht bis zu 192000 Hz. Beide Effekte können sich ausgleichen, so könnte man den Abstand zwischen den Mikrofonen theoretisch auf die Hälfte verkürzen, wenn man die Abtastfrequenz verdoppelt.

Wenn man die Präzision weiter erhöht, ist es interessant die theoretisch maximale Präzision bei bestimmten Mikrofonabstand und Abtastfrequenz zu kennen. Dies ist aber nicht ohne Weiteres möglich, denn die Winkel sind nahe 0° durch die zeitliche Verschiebung höher aufgelöst, als zum Rand hin zu -90° und 90°. Dies ist ersichtlich durch den arccos in der Formel zur Berechnung der Richtung zur Geräuschquelle(2.3).

Ein besseres Ergebnis in Bezug auf Geräusche aus größerer Entfernung könnten hochwertigere Mikrofone mit Phantomspeisung liefern. Diese benötigen allerdings eine zusätzliche Stromversorgung, sind größer und liegen in einer deutlich höheren Preisklasse. Letztlich sollten die Mikrofone darauf ausgelegt sein bis zu der typischen Distanz zu einem Werfer gut genug aufzuzeichnen.

Doggys Gestänge sollte vergrößert werden oder der Computer gegen einen kleineren ausgetauscht, sodass dieser wieder in ihm untergebracht werden kann.

Alternativ könnten die Mikrofone am Oberteil des Roboters befestigt werden, wo sie sich mitdrehen. Dies würde allerdings bedeuten, dass sich auch Doggys Öffnungswinkel für interessante Geräusche mitdreht, was derzeit nicht erwünscht ist, da Doggy bisher nur auf Werfer aus einer Richtung reagieren soll.

Im Rahmen dieser Arbeit wurden in dem ROS-Knoten Model zwei Parameter implementiert, durch die Doggy stärker auf Geräusche von vorne reagiert und sich weniger weit zur Seite dreht. Die Funktionalität dieser Parameter wurde überprüft, bei den Tests zu dieser Arbeit wurden sie jedoch nicht angewendet, da es hauptsächlich darum ging das korrekte Erkennen von Geräuschen zu testen und nicht Doggys Wirkung auf die Probanden. Es bleibt noch zu testen, ob Doggys Verhalten durch sie menschlicher wirkt.

Literaturverzeichnis

[1] Joseph L Horner und Peter D Gianino: *Phase-only matched filtering.* Applied optics, 23(6):812–816, 1984.

[2] Tim Laue, Oliver Birbach, Tobias Hammer und Udo Frese: *An Entertainment Robot for Playing Interactive Ball Games.* In: *RoboCup 2013: Robot World Cup XVII*, Seiten 171–182. Springer, 2014.

[3] John G Proakis: *Digital signal processing: principles algorithms and applications.* Pearson Education India, 2001.

[4] Fredrik Gustafsson und Fredrik Gunnarsson: *Positioning using time-difference of arrival measurements.* In: *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, Band 6, Seiten VI–553. IEEE, 2003.

[5] Mathias Buder: *Sprecherlokalisierung eines Sprechers im Raum unter Verwendung eines kugelförmigen Mikrofonarrays.* Masterarbeit, Hochschule für angewandte Wissenschaften Hamburg, 2013.

[6] Charles Knapp und G Clifford Carter: *The generalized correlation method for estimation of time delay.* Acoustics, Speech and Signal Processing, IEEE Transactions on, 24(4):320–327, 1976.

[7] Karl Dirk Kammeyer und Kristian Kroschel: *Digitale Signalverarbeitung: Filterung und Spektralanalyse mit MATLAB-Übungen.* Springer DE, 2009.

[8] Tilman Butz: *Fouriertransformation für Fußgänger.* Springer, 2009.

[9] Harry Nyquist: *Certain topics in telegraph transmission theory.* American Institute of Electrical Engineers, Transactions of the, 47(2):617–644, 1928.

[10] *Norio Ohga: Vom Kopilot zum Kommodore.* `http://www.handelsblatt.com/unternehmen/management/koepfe/norio-ohga-vom-kopilot-zum-kommodore/2957156.html`, 2008. Abgerufen am: 04.11.2014.

[11] *Struktur des digitalen Regelkreises.* `http://me-lrt.de/struktur-digitaler-regelkreis-signalweg-aliasing-modellierung`. Abgerufen am: 04.11.2014.

[12] Heiko Bülow und Andreas Birk: *Spectral registration of noisy sonar data for underwater 3D mapping.* Autonomous Robots, 30(3):307–331, 2011.

[13] Simon S Haykin: *Adaptive filter theory*. Pearson Education India, 2008.

[14] Norbert Wiener: *Extrapolation, interpolation, and smoothing of stationary time series*, Band 2. MIT press Cambridge, MA, 1949.

[15] ROBERT A Hinde: *Behavioural habituation*. Short-term changes in neural activity and behaviour. Cambridge University Press, Cambridge, Seiten 3–40, 1970.

[16] Takuya OOURA: *General Purpose FFT (Fast Fourier/Cosine/Sine Transform) Package*. `http://www.kurims.kyoto-u.ac.jp/~ooura/fft.html`, 1996. Abgerufen am: 10.11.2014.

[17] Steven G. Johnson und Matteo Frigo: *Implementing FFTs in Practice*. In: C. Sidney Burrus (Herausgeber): *Fast Fourier Transforms*, Kapitel 11. Connexions, Rice University, Houston TX, September 2008. `http://cnx.org/content/m16336/`.

[18] Jay Sachdev Crayzee Wulf, Jan Wedekind: *Serial Port Programming in C++*. `http://libserial.sourceforge.net`, 2014. Abgerufen am: 20.11.2014.

[19] Evangelischer Kindergarten Kelterplatz in Beuren: *Lieder auf dem Brunnenfest*. `https://www.youtube.com/watch?v=iZ5uChw102Y`, 2009. Abgerufen am: 17.11.2014.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich meine Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Bremen, den 2. Januar 2015