Technische Universität Hamburg-Harburg
Vision Systems

Prof. Dr.-Ing. R.-R. Grigat

Implementierung und Evaluation einer Pfeifendetektion für den NAO-Roboter

Bachelorarbeit

Arne Hasselbring

4. Oktober 2017

Eidesstattliche Erklärung

Ich, Arne Hasselbring, geboren am 03.12.1997 in Hamburg, versichere hiermit an Eides statt, dass ich diese von mir bei der Technischen Universität Hamburg-Harburg (TUHH) vorgelegte Bachelorarbeit selbstständig verfasst habe. Ich habe ausschließlich die angegebenen Quellen und Hilfsmittel benutzt.

Ort und Datum	Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Autonome mobile Roboter spielen in vielen Bereichen des Lebens eine immer wichtigere Rolle. Damit sie sich in einer Umgebung mit Menschen zurechtfinden können, müssen sie mit einer Vielzahl von Sensoren ausgestattet sein. Dazu zählen Kameras und darauf aufbauende Bildverarbeitungsalgorithmen, die dem Roboter den Sehsinn des Menschen ersetzen. Aber auch Hören soll ein Roboter können, z. B. um Anweisungen entgegenzunehmen. Obwohl mittlerweile hochqualitative Spracherkennungssysteme existieren, ist dies eine komplexe Angelegenheit. Eine alternative Möglichkeit zur Steuerung von Robotern sind einfache akustische Signale, wie z. B. Pfiffe.

Ein Bereich, in dem auch Menschen Pfeifen verwenden, sind Sportwettbewerbe. Im Fußball wird eine Pfeife vom Schiedsrichter eingesetzt, um den Spielern Entscheidungen mitzuteilen und das Spiel zu beginnen oder anzuhalten. Damit Roboter menschenähnlich Fußball spielen können, müssen sie demnach auch in der Lage sein, diese Signale akustisch zu verstehen.

In dieser Arbeit geht es um das spezielle Problem, mit einem autonomen fußballspielenden Roboter den Anpfiff des Schiedsrichters zu erkennen.

1.1 Motivation

In der Standard Platform League (SPL) des RoboCup ist seit 2015 der Anpfiff von den Robotern zu erkennen [Com15, S. 12]. War dies zunächst nur in Ausscheidungsspielen der Fall, werden seit 2016 alle Spiele mit einer Pfeife gestartet [Com16, S. 42]. Bereits 2014 wurde ein technischer Wettbewerb ausgetragen, bei dem die Verarbeitung von Audiosignalen auf dem NAO-Roboter erprobt wurde [Com14, S. 4-5].

Dabei führt das Nichterkennen der Pfeife dazu, dass der Roboter erst 15 s nach dem Anpfiff per W-LAN das Startsignal bekommt. Erkennt ein Roboter jedoch eine Pfeife vor dem tatsächlichen Anpfiff, z. B. durch Rufe von Zuschauern, und läuft infolgedessen von seiner Anstoßposition los, wird er bestraft, indem er die ersten 15 s nach Anpfiff

stehen bleiben muss. Die korrekte Erkennung der Pfeife kann somit großen Einfluss auf das Spielgeschehen haben, da sie darüber entscheidet, welches Team zuerst den Ball erreicht. Ein Anstoß mit Anpfiff findet zudem nicht nur am Anfang der Halbzeiten statt, sondern auch nach jedem Tor.

Die maschinelle Erkennung von Pfeifen wird auch in der Sportinformatik untersucht. So stellten Tjondronegoro, Chen und Pham [TCP03] ein System vor, das automatisch Zusammenschnitte besonderer Situationen von Aufnahmen von Fußballspielen und Schwimmwettbewerben erstellt.

1.2 Zielsetzung und Abgrenzung

Das Ziel dieser Arbeit ist es, einen Detektionsalgorithmus für Pfeifen zu entwickeln und ihn gegen andere Lösungen auszuwerten. Dazu muss zunächst eine Software entwickelt werden, die es ermöglicht, verschiedene Detektoren auf einem standardisierten Datensatz auszuführen. Der Datensatz muss aufgenommen und von Hand annotiert werden, um die Auswertung automatisch durchführen zu können. In der Auswertungssoftware sollen mehrere veröffentlichte Lösungen anderer SPL-Teams implementiert werden. Letztendlich soll eine weitere Methode im Rahmen dieser Arbeit entwickelt werden und mit den anderen verglichen werden.

Die in dieser Arbeit untersuchten Kriterien für einen Pfeifenerkennungsalgorithmus sind folgende:

Trefferquote Ein Pfeifendetektionsalgorithmus muss echte Pfeifen als solche erkennen können. Es ist zu erwarten, dass nicht jede Pfeife erkannt wird, z. B. weil zu leise gepfiffen wird oder sie durch Lärm übertönt wird. Trotzdem sollte die Anzahl erkannter Pfeifen möglichst hoch sein.

Fehldetektionshäufigkeit Fehldetektionen sind Ereignisse, bei denen ein Detektionsalgorithmus eine Pfeife meldet, obwohl in dem Moment auf dem eigenen Spielfeld nicht gepfiffen wurde. Die Häufigkeit davon muss möglichst gering sein, da ein Roboter, der vor dem eigentlichen Anpfiff losläuft, eine Zeitstrafe erhält.

Kalibrierungsaufwand Fast alle denkbaren Pfeifenerkennungsalgorithmen haben mehrere Parameter, also einstellbare Werte, die die Erkennung beeinflussen. Es ist dabei unpraktisch, wenn diese Parameter auf dem Wettbewerb oder sogar vor jedem Spiel angepasst werden müssen. Im schlimmsten Fall könnte für jede Pfeife ein unterschiedlicher Parametersatz benötigt werden. Welche Pfeife genau verwendet wird, kann sich allerdings theoretisch noch während eines Spiels ändern. Somit ist es wünschenswert, dass ein Detektor mit unterschiedlichen Pfeifen und unterschiedlichen Lärmbedingungen zurechtkommt, ohne dass dazu umfangreiche Parameteränderungen nötig sind.

Rechenaufwand Da die Audiosignalverarbeitung auf dem Roboter in Echtzeit stattfinden muss, ist der Rechenaufwand der Detektionsalgorithmen besonders relevant. Die Laufzeit sollte zumindest so klein sein, dass alle Audiodaten verarbeitet werden können, also die Verarbeitung von 1 s Audiodaten höchstens 1 s Rechenzeit benötigt. Auf dem Roboter werden allerdings gleichzeitig andere Algorithmen ausgeführt, sodass tatsächlich nur ein Bruchteil davon zur Verfügung steht.

Online-Fähigkeit Zusätzlich zu der Laufzeit des Algorithmus selbst ist auch relevant, wie viel Zeit vom Beginn des Pfeifengeräuschs bis zur Erkennung in der Software vergeht. Je mehr Audiodaten in die Entscheidung einbezogen werden, desto weniger Zeitvorteil bleibt dem Roboter, um tatsächlich zu handeln und z. B. eine Position in der gegnerischen Hälfte einzunehmen. Diese Reaktionszeit sollte also klein genug sein, dass sie im Vergleich zur Unsicherheit in der Laufgeschwindigkeit des Roboters vernachlässigbar ist.

Nicht untersucht wird in dieser Arbeit die Bestimmung der örtlichen Herkunft der Geräuschquelle. Da der NAO über mehrere Mikrofone verfügt, ist dies prinzipiell möglich, ohne sich auf die Lautstärke zu verlassen. Um die Pfeife auf dem eigenen Spielfeld von anderen benachbarten Feldern unterscheiden zu können, ist es auch wünschenswert. Allerdings ist die Lokalisierung mit den eng beieinander liegenden Mikrofonen schwierig, da die Ankunftszeiten der Signale sich kaum unterscheiden. Zusätzlich beeinflusst dann eventuell die Güte der Positionsschätzung des Roboters die Erkennung der Pfeife.

Außerdem verwenden einige SPL-Teams W-LAN-Kommunikation zwischen den Robotern, um gemeinsam zu entscheiden, ob ein Anpfiff stattgefunden hat. Damit wird sichergestellt, dass z. B. ein einzelner Roboter, der die Pfeife nicht hört, trotzdem losspielen kann, wenn die anderen Roboter die Pfeife erkannt haben. Diese Teamentscheidung ist von der Audiosignalverarbeitung vollständig unabhängig und ist auch nicht Teil dieser Arbeit, da sie im Team des Autors bereits vorhanden ist.

1.3 Aufbau dieser Arbeit

In Kapitel 2 werden die theoretischen Grundlagen der Audiosignalverarbeitung sowie die untersuchten Detektionsmethoden vorgestellt. Danach geht Kapitel 3 auf die praktische Implementierung des zur Auswertung verwendeten Softwaresystems ein. Im 4. Kapitel werden die Ergebnisse der Auswertung präsentiert und diskutiert. Zuletzt wird die Arbeit in Kapitel 5 zusammengefasst und es werden mögliche zukünftige Aufgaben benannt.

Kapitel 2

Grundlagen

Dieses Kapitel behandelt zunächst den Rahmen der Problemstellung, die Standard Platform League des RoboCup. Nachfolgend werden einige Grundlagen der Signalverarbeitung erläutert und die untersuchten Methoden zur Lösung des Problems vorgestellt.

2.1 RoboCup

Der RoboCup ist ein Anfang der 1990er Jahre erdachter und 1997 erstmals durchgeführter [Kit98] Wettbewerb, der die Vision verfolgt, dass bis Mitte des 21. Jahrhunderts ein autonomes Roboterfußballteam nach offiziellen Regeln gegen den menschlichen Fußballweltmeister gewinnen kann [KA00]. Durch dieses Ziel sollen Innovationen im Bereich der Robotik und der künstlichen Intelligenz einen öffentlichkeitswirksamen Rahmen bekommen. Nach dem Wettbewerb findet eine Konferenz statt, auf der die beteiligten Wissenschaftler ihre Erkenntnisse austauschen.

Neben den Fußballligen umfasst der RoboCup mittlerweile weitere Anwendungsbereiche der Robotik: In der RoboCupRescue-Division wird an Robotern gearbeitet, die Rettungseinsätze durchführen können. Hier ist teilweise auch Fernsteuerung zulässig. Die RoboCup@Home-Ligen forschen an Servicerobotern, die Tätigkeiten im Haushalt verrichten. Im RoboCupIndustrial-Bereich geht es um Roboter im Umfeld von Produktion und Logistik.

2.1.1 Standard Platform League

Die Standard Platform League hat die Besonderheit, dass alle Teams die gleiche Roboterplattform einsetzen müssen. Dadurch liegt der Fokus mehr auf der Forschung im Bereich der künstlichen Intelligenz als in den anderen humanoiden Ligen, die sich zusätzlich noch mit der Konstruktion der Roboter beschäftigen.

Der NAO-Roboter wird seit 2006 von Aldebaran Robotics (inzwischen SoftBank Robotics) in verschiedenen Hardware-Revisionen hergestellt. Die bisher neueste Generation wurde 2014 vorgestellt, ist gut 57 cm groß und wiegt 5.4 kg [Rob17]. Als Prozessor kommt ein Intel Atom Z530 mit 1.6 GHz Taktfrequenz und 1 GiB Arbeitsspeicher zum Einsatz. Der NAO besitzt 25 Freiheitsgrade, zwei Kameras, eine inertiale Messeinheit und einige weitere Sensoren. Die Audiohardware besteht aus zwei Lautsprechern an den Seiten des Kopfes und vier Mikrofonen auf der Kopfoberseite (s. Abb. 2.1). Laut Spezifikation [Rob17] umfassen diese einen Frequenzbereich von 150 Hz bis 12 kHz. Seit 2008 ist der NAO, zunächst noch neben dem älteren Roboterhund AIBO, die Standardplattform der RoboCup-Fußballwettbewerbe.

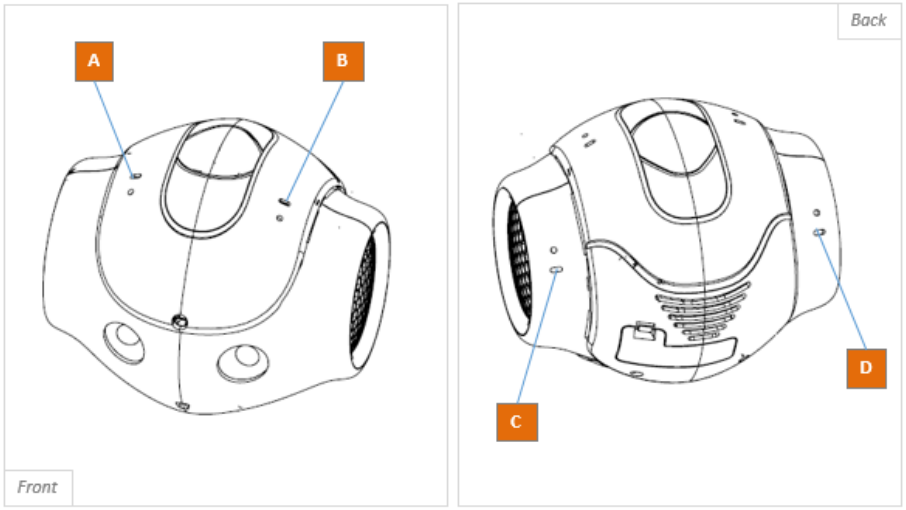

Abbildung 2.1: *Die Positionen der Mikrofone des NAOs [Rob17]*

Es spielen Mannschaften von jeweils fünf Spielern auf einem 9 m × 6 m großen Feld zwei 10 min lange Halbzeiten [Com16]. Die Roboter müssen dabei komplett autonom handeln. Sie dürfen allerdings mit einer beschränkten Datenrate untereinander über W-LAN kommunizieren. Außerdem bekommen sie Veränderungen des Spielzustandes, z. B. gefallene Tore oder Zeitstrafen für einzelne Spieler, über ein von einem Menschen bedientes Schiedsrichterprogramm, den sog. *GameController*, mitgeteilt. Die Spielzustände sind *Initial*, *Ready*, *Set*, *Playing* und *Finished*. Der *Initial*-Zustand besteht vor dem Spiel, wenn die Roboter an die Seitenlinien zum Einlaufen gestellt werden. Der Zustand wird auf *Ready* umgeschaltet, damit die Roboter auf ihre Anstoßpositionen in der eigenen Hälfte laufen. Dieser Zustand tritt außerdem nach jedem Tor auf. Nach Ablauf von 45 s wird automatisch in den *Set*-Zustand übergegangen. In diesem Zustand dürfen sich die Roboter nicht von der Stelle bewegen. Der Schiedsrichter legt den Ball auf den Anstoßpunkt und in bestimmten Fällen werden Roboter zu ihrem Nachteil manuell umplatziert. Während dieser Zeit muss die Pfeifenerkennung ausgeführt werden,

denn durch den Anpfiff geht das Spiel in den *Playing*-Zustand über. In allen anderen Zuständen kann die Pfeifenerkennung deaktiviert bleiben, sodass die dafür verwendeten Rechenressourcen nicht verschwendet werden. Technisch ist dieser Übergang so realisiert, dass der *GameController* den Beginn des *Playing*-Zustandes erst dann per W-LAN mitteilt, wenn 15 s im *Playing*-Zustand vergangen sind. Dadurch haben auch Roboter, die die Pfeife nicht hören, nach einer gewissen Zeit die Chance, am Spiel teilzunehmen. Das Ende einer Halbzeit wird den Robotern durch Übergang in den *Finished*-Zustand mitgeteilt.

Bewegt sich ein Roboter noch im *Set*-Zustand von seiner Position fort, wird gegen ihn die Strafe *Illegal Motion in Set* verhängt. Die Strafe besteht darin, dass der Roboter an seine vorige Position zurückgestellt wird und erst dann losspielen darf, wenn 15 s im *Playing*-Zustand vergangen sind. Das Auftreten von *Illegal Motion in Set* ist meistens gleichwertig zu einer Fehldetektion der Pfeifenerkennung. Es gibt allerdings auch den Fall, dass auf einem benachbarten Spielfeld ein Anpfiff stattfindet. Dies ist in den Regeln explizit genannt, wird dort aber als „unwahrscheinlich und Pech“ [Com16, S. 12] bezeichnet. Die Erfahrung aus vergangenen Wettbewerben zeigt jedoch, dass dies immer wieder vorkommt. Die Roboter werden dann trotzdem bestraft, wenn sie loslaufen.

Es ist zu bemerken, dass obwohl *Illegal Motion in Set* und Nichterkennen des Anpfiffs scheinbar die gleichen Konsequenzen — 15 s Verzögerung — haben, dies nicht ganz der Fall ist. Das Zurückstellen des Roboters nach *Illegal Motion in Set* kann große Konsequenzen auf die Positionsschätzung des Roboters haben. Selbst wenn dieser Fall in der Selbstlokalisierung des Roboters behandelt wird, zeigt die Praxis, dass die Schiedsrichterassistenten die Positionierung häufig nicht korrekt durchführen.

Die Regeln [Com16] erlauben dem Schiedsrichter die Verwendung einer beliebigen Sport-Pfeife. Das bedeutet, dass die Detektoren sich nicht auf eine exakte, immer gleiche Grundfrequenz des Tons verlassen können. Es gibt insbesondere keine Vorschrift darüber, ob eine Trillerpfeife oder eine Pfeife ohne Kugel verwendet wird.

Bei der Anstoßsituation ist weiterhin zu beachten, dass die Mannschaft, die nicht Anstoß hat, den Mittelkreis erst nach 10 s oder einer Ballberührung des Gegners betreten darf. Hört also die angreifende Mannschaft die Pfeife nicht, bleiben der verteidigenden nur 5 s Zeit, ungehindert den Ball zu bewegen. In dieser Zeit kann trotzdem schon an mindestens einem Roboter vorbeigespielt werden. Hört die Mannschaft, die Anstoß hat, als einzige die Pfeife, können die resultierenden 15 s bereits für ein Tor ausreichen.

2.2 Grundlagen der Signalverarbeitung

Die Entstehung des Pfeifentons, die Ausbreitung des Schalls und die Funktionsweise des Mikrofons werden in dieser Arbeit nicht behandelt. Es wird hier davon ausgegangen, dass ein Mikrofon die Schwingungen des Schalls als Spannung bereitstellt. Diese

Spannung wird als analoges, d. h. sowohl im Zeit- als auch im Wertebereich kontinuierliches Signal betrachtet.

Zur Verarbeitung in einem Computer muss das Signal in beiden Bereichen diskretisiert werden. Durch Abtastung wird es zunächst ein zeitdiskretes Signal, welches dann von einem Analog-Digitalwandler quantisiert wird. Die Abtastrate und die Auflösung im Wertebereich sind dabei Parameter, die die Qualität des digitalen Signals bestimmen. Die resultierende Folge von Signalwerten in Abhängigkeit von der diskreten Zeit ist die Eingabe der hier betrachteten Signalverarbeitungsalgorithmen.

Neben der Darstellung im Zeitbereich gibt es auch noch die Möglichkeit, das Signal in seine Frequenzanteile zu zerlegen. Im Zusammenhang mit Audiosignalen ist dies deshalb sinnvoll, weil Töne sich aus harmonischen Schwingungen verschiedener Frequenzen zusammensetzen. Die Grundfrequenz bestimmt dabei die Tonhöhe, ggf. weitere vorhandene Frequenzanteile bei Vielfachen der Grundfrequenz, die sog. Obertöne, bestimmen den Klang.

Die Frequenzanteile des Signals können durch die diskrete Fourier-Transformation (DFT) berechnet werden [PLJ15, S. 254–265]. Zur Transformation von $N \in \mathbb{N}$ Werten im Zeitbereich lautet die Formel

$$X[k] = \sum_{j=0}^{N-1} x[j] \cdot \exp(-2\pi i \cdot \frac{jk}{N}) \qquad \forall k = 0, \ldots, N-1.$$

Die Folge der $X[k]$ wird als das Spektrum des Signals bezeichnet. Dieses ist im Allgemeinen komplex, wobei der Betrag eines $X[k]$ die Amplitude und die Phase die Verschiebung der entsprechenden Schwingung im Zeitbereich ist.

Das Spektrum ist periodisch mit der Periodendauer N, d. h. $X[k+N] = X[k]$, da es eine Summe von Schwingungen bei ganzzahligen Vielfachen der Frequenz $\frac{1}{N}$ ist. Da bei Audiodaten die $x[j]$ reell sind, ist das Spektrum außerdem konjugiert gerade, weil die komplexen Schwingungen konjugiert gerade sind. Aufgrund dieser beiden Eigenschaften ist der Teil des Spektrums über $\frac{N}{2}$ redundant; das Spektrum enthält also nur $\lfloor \frac{N}{2} \rfloor + 1$ unterschiedliche Werte. Wichtig ist außerdem, dass das Spektrum tatsächlich die N-periodische Fortsetzung des Signals im Zeitbereich beschreibt.

Die Anzahl der an der DFT beteiligten Werte im Zeitbereich, auch DFT-Größe genannt, bestimmt die Auflösung im Frequenzbereich: Um einen Index im diskreten Spektrum in die zugehörige Frequenz umzuwandeln, muss er mit dem Quotienten aus Abtastrate und DFT-Größe multipliziert werden.

Eine einzelne DFT sagt nichts über die zeitliche Veränderung der Frequenzanteile des Signals aus. Dies ist aber zur Pfeifenerkennung notwendig, da der Pfiff nur einen kurzen Teil des Signals einnimmt. Außerdem steht während der Ausführung der Pfeifendetektion noch gar nicht das gesamte Signal zur Verfügung. Daher verwenden alle vorgestellten Pfeifendetektionsmethoden die Kurzzeit-Fourier-Transformation (STFT). Eine STFT besteht aus vielen zeitlich aufeinanderfolgenden DFTs, die jeweils nur einen

kleinen Ausschnitt des Signals im Zeitbereich umfassen [PLJ15, S. 431]. Durch diese Ausschnitte kann das Vorhandensein von Frequenzanteilen zu bestimmten Zeitpunkten festgestellt werden. Je größer das Beobachtungsfenster ist, desto höher ist die Frequenzauflösung, allerdings verringert sich damit die Auflösung im Zeitbereich.

Bei der DFT kommt es zu dem unerwünschten Effekt, dass das Spektrum Frequenzanteile enthält, die scheinbar nicht im Signal vorhanden sein sollten. Diese kommen daher, dass das Spektrum eigentlich zur N-periodischen Fortsetzung des Signals gehört, die an den Rändern Sprünge hat. Um diesen sog. Leck-Effekt zu verringern, kann das Signal im Zeitbereich elementweise mit einer Fensterfunktion multipliziert werden, die zu den Rändern hin verschwindet. Dadurch werden die unerwünschten Frequenzanteile abgeschwächt, gleichzeitig sinkt aber die Frequenzauflösung. Die Auswahl guter Fensterfunktionen ist ein ganz eigenes Problem, das ausführlich z. B. in [Har78] behandelt wird. Der Einfachheit halber verwenden alle hier vorgestellten Methoden entweder ein Rechteckfenster — d. h. das Signal wird einfach ausgeschnitten und transformiert — oder ein Hann-Fenster.

2.3 Detektionsmethoden

In diesem Abschnitt werden die Detektionsmethoden vorgestellt, die verglichen werden. Die ersten vier davon sind von anderen Teams aus der SPL entwickelt und im Quelltext oder in Form schriftlicher Dokumentation veröffentlicht worden. Sie werden in dieser Arbeit mit den Teamnamen identifiziert. Die letzte Methode wurde im Rahmen dieser Arbeit entwickelt.

Alle hier behandelten Detektoren arbeiten in Zeitschritten, d. h. sie lesen eine bestimmte Menge von Audiosamples ein, stellen damit ihre Berechnungen an und treffen eine Entscheidung, ob eine Detektion gemeldet wird. Dabei kann die Entscheidung, ob ein Pfiff gemeldet wird, von einem internen Zustand des Detektors abhängen. So prüfen einige Detektoren eine Mindestlänge des Pfiffs, indem sie den Pfiff erst melden, wenn eine bestimmte Anzahl von Spektren hintereinander die Pfeifenkriterien erfüllt hat.

2.3.1 Bembelbots

Das SPL-Team aus Frankfurt verfolgt den Ansatz, zu überprüfen, ob die Frequenz mit der höchsten Amplitude lange genug in einem bestimmten Bereich liegt [Bem16]. Dieser Detektor hält also Zustand über mehrere verarbeitete Spektren, um seine Entscheidung zu treffen.

Für jedes einzelne Spektrum werden zunächst so viele Samples gelesen, dass ein reeller Puffer für die DFT voll ist. Die aufeinanderfolgenden Puffer überlappen sich zeitlich nicht, d. h. jedes Audiosample ist Teil von genau einer DFT. Nach Durchführung

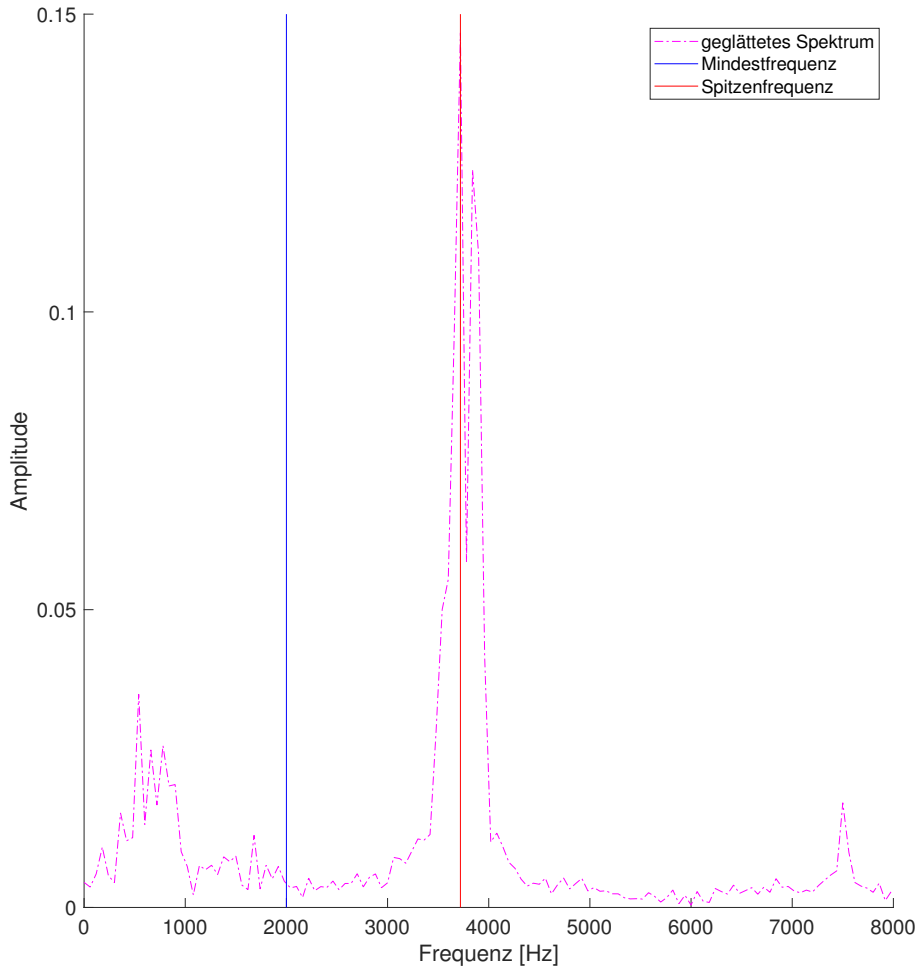

Abbildung 2.2: *Darstellung der Mindestfrequenz f_{min} und der Spitzenfrequenz f_{peak} im Spektrum einer Pfeife. Die Frequenzachse ist nach rechts abgeschnitten, da die höheren Frequenzen für den Algorithmus nicht relevant sind.*

der DFT werden die Beträge des komplexen Spektrums genommen, um das Amplitudenspektrum zu erhalten. Es wird keine spezielle Fensterfunktion anstelle des Rechteckfensters zur Abschwächung des Leck-Effekts benutzt. Stattdessen wird das Spektrum geglättet, indem n_{smooth} aufeinanderfolgende Amplituden gemittelt werden und somit ein Spektrum der Länge $n_{\text{DFT}}/n_{\text{smooth}}$ entsteht. In diesem geglätteten Spektrum wird ohne weitere Einschränkungen die Frequenz f_{peak} mit der höchsten Amplitude bestimmt. Das Spektrum wird nun als möglicher Teil einer Pfeife angenommen, wenn f_{peak} einen im Voraus wählbaren Schwellwert f_{min} überschreitet. Die Idee dabei ist, dass die meisten Hintergrundgeräusche sich in einem Frequenzbereich befinden, der unterhalb der Grundfrequenz der meisten Pfeifen ist. Dies ist in Abb. 2.2 dargestellt.

An dieser Stelle wird der bisherige Zustand des Detektors in Form des aktuellen Pfeifenkandidaten einbezogen. Ein Pfeifenkandidat enthält die zeitliche Länge, die er bereits andauert, und die höchste Amplitude im Zeitbereich, die währenddessen aufgetreten ist. Falls noch kein Kandidat vorhanden ist, aber das aktuelle Spektrum als Teil einer Pfeife erkannt wurde, wird ein neuer Kandidat mit der Dauer eines Puffers und der Maximalamplitude im aktuellen Puffer angelegt. Wenn schon ein Kandidat vorhanden ist und das aktuelle Spektrum als Teil einer Pfeife erkannt wurde, wird der Kandidat um die Pufferlänge verlängert und ggf. die Maximalamplitude angepasst. Sollte der Kandidat dadurch die Mindestdauer t_{min} und die Mindestamplitude a_{min} überschreiten, wird ein Pfiff gemeldet. Falls im aktuellen Spektrum f_{min} nicht überschritten wurde, wird der ggf. vorhandene Kandidat gelöscht.

2.3.2 HULKs

Das SPL-Team des Autors, die Hamburg Ultra Legendary Kickers (HULKs) von der Technischen Universität Hamburg-Harburg, hat seit kurz vor dem RoboCup 2016 eine eigene Pfeifendetektion [Rie+16, S. 6]. Die Grundidee ist es, die Energie des Signals in einem vorkonfigurierten Pfeifenband im Verhältnis zum Rest des Signals zu untersuchen. Die Entscheidung, ob eine Pfeife gemeldet wird, wird für jedes Spektrum einzeln getroffen, d. h. der Detektor hält keinen Zustand.

Der Detektor liest einen Eingangspuffer für die DFT ein und führt ohne weitere Multiplikation mit einer Fensterfunktion die DFT durch. Aus dem erhaltenen Spektrum wird die spektrale Energiedichte bestimmt, indem die Beträge des Spektrums quadriert werden [PLJ15, S. 236]. Durch Summierung der Energiedichte über das Pfeifenband, das durch eine untere Frequenz f_{min} und eine obere Frequenz f_{max} charakterisiert wird, wird die sog. Pfeifenenergie berechnet.

Als Stoppband wird lediglich der Frequenzbereich oberhalb des Pfeifenbandes benutzt; der Bereich unterhalb des Pfeifenbandes wird nicht berücksichtigt. Die Summe über die Energiedichte im Stoppband ergibt die Stoppenergie.

Zur Entscheidung wird nun lediglich der Quotient aus Pfeifenenergie und Stoppenergie bestimmt und mit einem Schwellwert $c_{\text{threshold}}$ verglichen. Ist der Quotient größer als der Schwellwert, so wird die Pfeife sofort gemeldet, ohne dass die Entscheidung über mehrere Spektren gefiltert wird.

2.3.3 Nao Devils Dortmund

Der Detektor des SPL-Teams der TU Dortmund [Dor16] basiert darauf, dass Pfeifen keinen reinen Sinuston erzeugen, sondern in geringem Maße auch Obertöne produzieren.

Die Nao Devils verwenden überlappende Signale im Zeitbereich. Das bedeutet, dass jeder Eingabepuffer einer DFT bis zur Hälfte Werte enthält, die schon in der letzten

DFT benutzt wurden. Vor der Fourier-Transformation wird bei diesem Detektor das Eingangssignal mit einem Hann-Fenster multipliziert.

Algorithmus 1 Der Hauptteil des Detektionsalgorithmus der Nao Devils Dortmund

```
1:  function IsWhistle(amplitudes)
2:      whistleBegin ← FrequencyToIndex(f_min)
3:      whistleEnd ← FrequencyToIndex(f_max)
4:      peakPos ← arg max_{whistleBegin≤i≤whistleEnd} amplitudes[i]
5:      if amplitudes[peakPos] < a_0 then
6:          return false
7:      end if
8:      peak1Pos ← arg max_{c_1·peakPos≤i≤c_2·peakPos} amplitudes[i]
9:      if amplitudes[peak1Pos] < a_1 then
10:         return false
11:     end if
12:     peak2Pos ← arg max_{c_3·peakPos≤i≤c_4·peakPos} amplitudes[i]
13:     if amplitudes[peak2Pos] < a_2 then
14:         return false
15:     end if
16:     return true
17: end function
```

Es wird zunächst im Band zwischen $f_{\min}$ und $f_{\max}$ die Frequenz mit der höchsten Amplitude im Spektrum ermittelt. Überschreitet die Amplitude dort den Schwellwert a_0 nicht, wird das Spektrum sofort abgelehnt. Andernfalls wird im Bereich zwischen dem c_1-fachen und dem c_2-fachen der Spitzenfrequenz die höchste Amplitude bestimmt und mit einem weiteren Schwellwert a_1 verglichen. c_1 und c_2 liegen dabei wenig unter bzw. über 2, sodass damit der 1. Oberton getroffen werden soll. Wird a_1 dabei unterschritten, wird das Spektrum auch abgelehnt. Dasselbe wird für den 2. Oberton mit den Faktoren c_3 und c_4 und dem Schwellwert a_2 durchgeführt. Wenn auch dieser Schwellwert überschritten wird, gilt das Spektrum als möglicher Teil eines Pfiffs. Das Vorgehen ist als Pseudocode in Algorithmus 1 beschrieben.

Nachdem für ein einzelnes Spektrum entschieden wurde, ob es zu einer Pfeife gehören könnte, wird überprüft, ob kurz zuvor genug Spektren akzeptiert wurden. Ein Pfiff muss mindestens n_1 Spektren lang sein, wobei maximal n_2 einzelne Spektren fehlen dürfen. Wenn der Pfiff lang genug ist, wird er gemeldet.

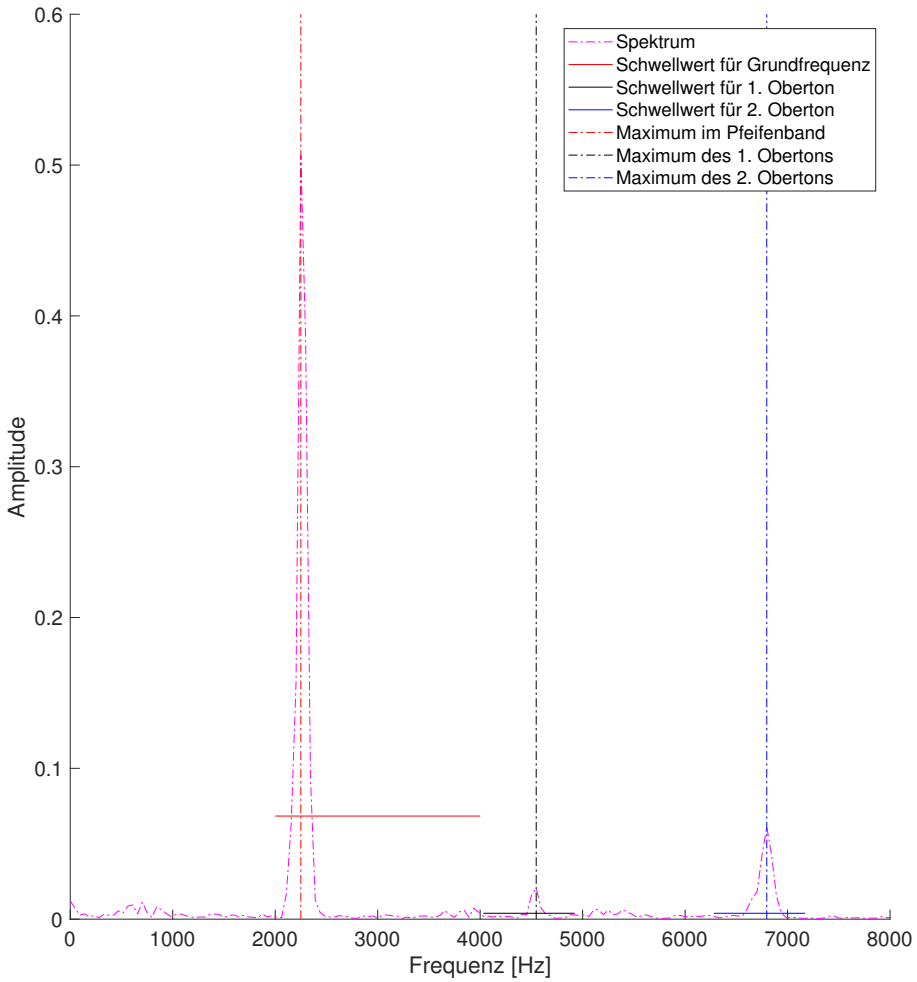

Abbildung 2.3: *Veranschaulichung des Nao-Devils-Algorithmus auf dem Amplitudenspektrum einer Pfeife. Die horizontalen Linien überdecken die Bereiche, in denen nach Maxima gesucht wird. Die Frequenzachse ist nach rechts abgeschnitten, da die höheren Frequenzen für den Algorithmus nicht relevant sind.*

2.3.4 UNSW Sydney

Das SPL-Team der University of New South Wales (UNSW) in Sydney, Australien, verwendet eine Methode [Syd15; Hal+15; Syd16], die Schwellwerte anhand statistischer Größen über dem Amplitudenspektrum bestimmt. Sie geht auf die veröffentlichte Pfeifenerkennung des Teams Austrian Kangaroos der TU Wien von 2014 zurück [Ham14; HS15].

Auch dieser Detektor beginnt damit, auf nicht überlappenden, konstant großen Puffern eine DFT durchzuführen und vom komplexen Spektrum die Beträge zu bilden. Von den Beträgen wird zunächst der Mittelwert μ und die Standardabweichung σ bestimmt.

Kernstück des Algorithmus ist die dynamische Bestimmung des Pfeifenbandes. Dieses wird innerhalb eines voreingestellten Bereichs zwischen f_{min} und f_{max} gesucht. Der Bereich wird dazu in n_{bucket} sog. Buckets, d. h. gleich große Intervalle im Frequenzbereich, eingeteilt. Von den Rändern des Bereichs wird dann innerhalb jedes Buckets die mittlere Amplitude berechnet und mit dem Schwellwert $\mu + c_{\text{background}} \cdot \sigma$ verglichen. Sobald dieser überschritten wird, wird die Suche beendet und die äußere Grenze des Buckets als Grenze des Pfeifenbandes festgelegt. Dieses Verfahren wird genau in Algorithmus 2 beschrieben.

Algorithmus 2 Die dynamische Bestimmung des Pfeifenbandes von UNSW Sydney

```
1:  function FINDWHISTLEBAND(spectrum, μ, σ)
2:      begin ← FREQUENCYTOINDEX(f_min)
3:      end ← FREQUENCYTOINDEX(f_max)
4:      growSize ← (end − begin)/n_bucket
5:      a_background ← μ + c_background · σ
6:      for i ← 0; i < n_bucket; i ← i + 1 do
7:          μ_bucket ← (Σ_{k=begin}^{begin+growSize−1} spectrum[k])/growSize
8:          if μ_bucket < a_background then
9:              begin ← begin + growSize
10:         else
11:             break
12:         end if
13:     end for
14:     for i ← 0; i < n_bucket; i ← i + 1 do
15:         μ_bucket ← (Σ_{k=end−growSize}^{end−1} spectrum[k])/growSize
16:         if μ_bucket < a_background then
17:             end ← end − growSize
18:         else
19:             break
20:         end if
21:     end for
22:     return (begin, end)
23: end function
```

Wenn kein einziger Bucket den Schwellwert überschreitet, wird angenommen, dass in dem Spektrum kein Pfiff vorhanden ist. Ansonsten wird der Mittelwert der Amplitude im Pfeifenband μ_{filtered} berechnet. Dieser wird mit dem Pfeifenschwellwert $a_{\text{whistle}} =$

$\mu + c_{\text{whistle}} \cdot \sigma$ verglichen. Hier unterscheiden sich die Versionen von 2015 und 2016: 2015 wurde zusätzlich noch ein Mechanismus verwendet, der überprüft, dass die Amplitude aktuell höher als in der Zeit vorher ist. Dazu werden in einem Ringpuffer μ und σ aller Spektren der vergangenen Sekunde gespeichert. Von beiden wird jeweils der Median berechnet und $\mu_{\text{median}} + c_{\text{temporal}} \cdot \sigma_{\text{median}}$ anstelle des ursprünglichen a_{whistle} verwendet, wenn es dieses überschreitet. Dieser Mechanismus wurde 2016 aus unbekannten Gründen entfernt. In beiden Fällen wird das Spektrum akzeptiert, wenn μ_{filtered} größer als a_{whistle} ist.

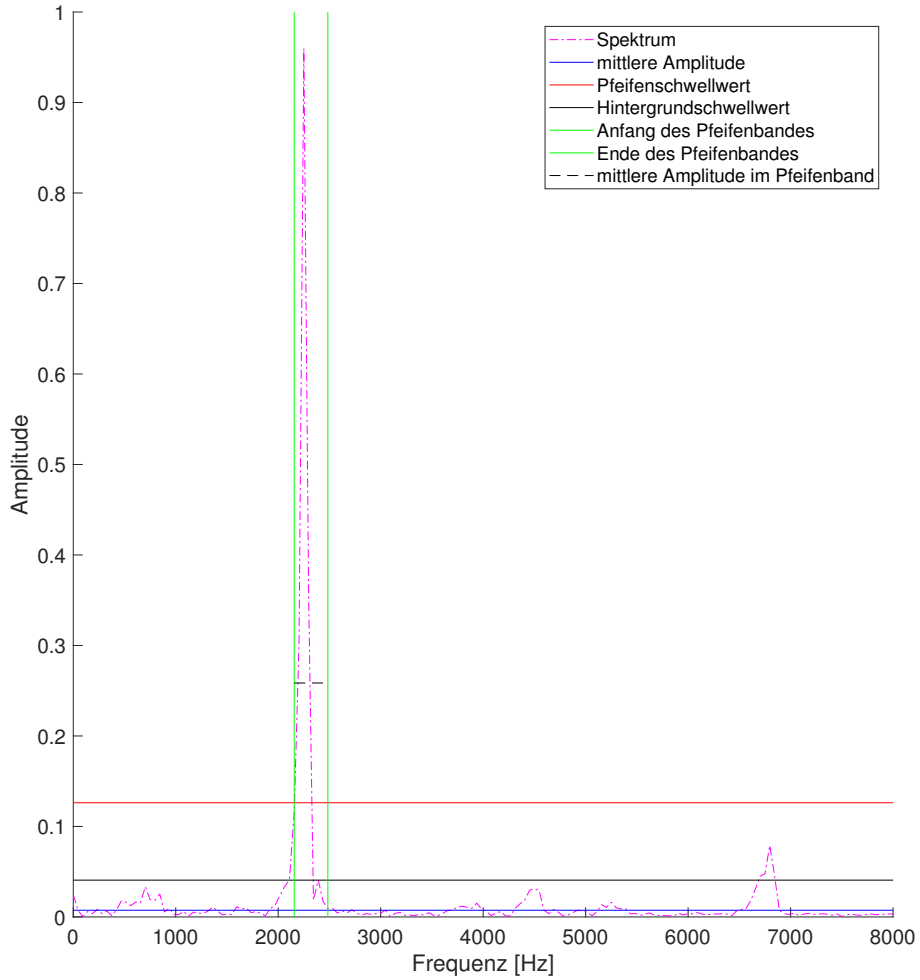

Abbildung 2.4: *Veranschaulichung des UNSW-Algorithmus auf dem Amplitudenspektrum einer Pfeife. Die Schätzung des Pfeifenbandes ist durch die grünen vertikalen Linien eingezeichnet.*

Nach der Entscheidung für das aktuelle Spektrum findet auch hier noch eine Filterung über die Zeit statt. Die Pfeife wird erst gemeldet, wenn Spektren einer äquiva-

lenten Zeit von t_1 akzeptiert wurden. Sie darf dabei maximal von aufeinanderfolgenden Spektren der Länge t_2 unterbrochen werden, ansonsten wird der Pfeifenzustand zurückgesetzt.

2.3.5 Eigene Methode

Das Ziel für die selbst entwickelte Methode war, Merkmale aus dem Spektrum zu extrahieren und anschließend mit einem maschinell gelernten Klassifizierer zu bewerten. Als Klassifizierer wird ein künstliches neuronales Netz benutzt. In der Frühphase des Projekts wurde auch ein Entscheidungsbaum getestet, dann aber gegen das neuronale Netz getauscht.

Der Detektor beginnt damit, Samples für eine DFT einzulesen, mit einem Hann-Fenster zu multiplizieren und zu transformieren. Anschließend wird durch Bildung der Beträge aus dem komplexen Spektrum das Amplitudenspektrum berechnet. Von hier teilt sich der Detektor in zwei Teile auf; die Berechnung des Merkmalsvektors und die Klassifizierung durch das künstliche neuronale Netz.

2.3.5.1 Merkmale

Die Merkmale basieren auf einer Einteilung des Frequenzbereichs in vier Bänder. Das 1. Pfeifenband soll die Grundfrequenz der Pfeife enthalten; das 2. Pfeifenband den 1. Oberton der Pfeife. Zwischen 1. und 2. Pfeifenband liegt das 1. Stoppband; über dem 2. Pfeifenband das 2. Stoppband. Die Idee ist, dass die Stoppbänder keine oder nur wenig Signalanteile der Pfeife enthalten. Das Frequenzband unterhalb des 1. Pfeifenbandes ist für den Detektor nicht relevant.

Als erstes wird in einem konfigurierbaren Frequenzbereich zwischen f_{min} und f_{max} die Frequenz mit der höchsten Amplitude f_{peak} gesucht. Es wird davon ausgegangen, dass es sich hierbei um die Grundfrequenz der Pfeife handelt. Überschreitet die Amplitude dort einen konfigurierbaren Schwellwert a_{min} nicht, wird das Spektrum bereits ohne weitere Klassifizierung verworfen und keine Pfeife gemeldet. Dieser Schwellwert kann empirisch so festgelegt werden, dass er von allen Pfeifen in mindestens einem Spektrum überschritten wird.

Anschließend wird von f_{peak} aus zu beiden Seiten der Anfang und das Ende des Pfeifenbandes gesucht. Die Suche wird erfolgreich beendet, wenn die Amplitude um den Faktor c_{minmax} kleiner als die Maximalamplitude ist. Falls dies bis zu einer bestimmten Entfernung von f_{peak} nicht eingetroffen ist, wird die Frequenz mit der bis dahin kleinsten Amplitude als Grenze genommen. Die so bestimmte untere und obere Grenze legen das 1. Pfeifenband fest.

Das 2. Pfeifenband wird zentriert um $2 \cdot f_{\text{peak}}$ gelegt, wobei als Breite die Hälfte des 1. Pfeifenbandes angenommen wird. Durch die Grenzen der Pfeifenbänder sind auch die Stoppbänder festgelegt. Die Einteilung ist in Abb. 2.5 zu sehen.

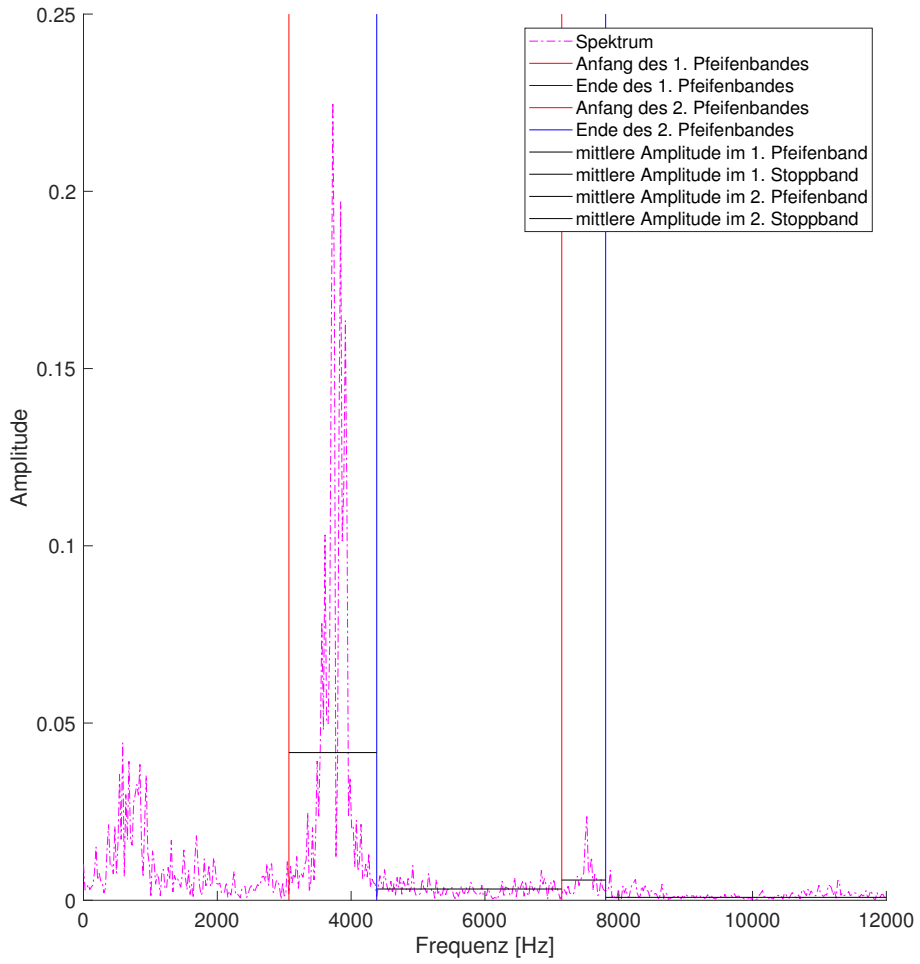

Abbildung 2.5: *Die bestimmten Bänder mit ihren mittleren Amplituden. Die Frequenzachse ist nach rechts abgeschnitten, da dort kaum noch Frequenzanteile des Signals liegen.*

Für jedes dieser vier Bänder wird die mittlere Amplitude bestimmt. Die sechs Merkmale, die dem Klassifizierer zur Verfügung gestellt werden, sind folgende:

- die Amplitude bei f_{peak}
- der Quotient der Durchschnittsamplituden im 1. Pfeifenband und 1. Stoppband
- der Quotient der Durchschnittsamplituden im 2. Pfeifenband und 1. Stoppband
- der Quotient der Summen der durchschnittlichen Pfeifenbandamplituden und der durchschnittlichen Stoppbandamplituden
- der Quotient der Durchschnittsamplituden im 1. Stoppband und 2. Stoppband

- der Quotient der Durchschnittsamplituden im 1. Pfeifenband und 2. Pfeifenband

2.3.5.2 Künstliches Neuronales Netz

Künstliche neuronale Netze sind eine Art biologisch inspiriertes Rechenmodell [Roj96]. Sie bestehen aus der Verbindung einzelner abstrahierter Neuronen, wobei jedes Neuron aus mehreren Eingabewerten die Summe berechnet, darauf eine nichtlineare Aktivierungsfunktion anwendet und das Ergebnis an seinem Ausgang bereitstellt. Die Parameter, über die sich ein künstliches neuronales Netz an die zu berechnende Funktion anpasst, sind die Gewichte der Verbindungen zwischen den Neuronen. Diese werden von einem Optimierer so bestimmt, dass der mittlere quadrierte Ausgabefehler minimal wird. Der Vorgang der Optimierung wird als Training bezeichnet. Einige Neuronen speisen die Eingabemerkmale in das Netz, während an anderen die berechnete Gesamtausgabe des Netzes abgegriffen wird. Die restlichen Neuronen werden als verdeckte Neuronen bezeichnet.

Bei diesem Problem ist die zu berechnende Funktion die Abbildung eines Merkmalsvektors auf 1, wenn die Merkmale zu einem Pfiff gehören, und 0 andernfalls. Die Eingabe des neuronalen Netzes ist also der Merkmalsvektor und die Ausgabe eine Zahl $\in [0, 1]$. Um von dieser kontinuierlichen Zahl zu einer binären Klassifizierung zu kommen, wird ein Schwellwert benutzt. Alle Ausgabewerte über diesem Schwellwert werden zur Pfeifenklasse gezählt.

Das hier verwendete Netz besteht aus 6 Eingabeneuronen — eins für jedes Merkmal —, 8 verdeckten Neuronen und einem Ausgabeneuron. Die verdeckten Neuronen benutzen als Aktivierungsfunktion den tanh, das Ausgabeneuron die Sigmoidfunktion.

Bevor die Merkmale in das neuronale Netz geführt werden, werden sie normalisiert. Dazu werden während des Trainings von allen Merkmalsvektoren komponentenweise Mittel und Standardabweichung berechnet. Bei der Klassifizierung wird dann vom Merkmalsvektor das Mittel subtrahiert und durch die Standardabweichung dividiert. Es stellte sich heraus, dass dies positiven Einfluss auf die Ergebnisse hat.

Für das Training wurde ein zufällig ausgewählter balancierter Datensatz zusammengestellt, d. h. er enthält dieselbe Anzahl positiver wie negativer Trainingsbeispiele. Dies soll verhindern, dass der Klassifizierer häufiger die Klasse mit mehr Elementen ausgibt, nur weil dadurch im Durchschnitt der Ausgabefehler geringer wird. Die positiven Trainingsbeispiele stammen von Spektren, die mindestens zur Hälfte zu einem Pfiff gehören. Alle anderen Spektren sind negative Trainingsbeispiele. Auf eine Validierung musste beim Training leider verzichtet werden. Dies liegt an der sehr geringen Menge an Daten. Aufgrund der sehr kleinen Struktur des Netzes ist Überanpassung aber eher unwahrscheinlich.

2.3.6 Weitere Methoden

Es gibt noch weitere SPL-Teams, die ihre Methoden veröffentlicht haben. Diese wurden aus Gründen des Umfangs in dieser Arbeit nicht implementiert und werden daher nur kurz vorgestellt.

Das SPL-Team B-Human aus Bremen benutzt zur Detektion der Pfeife die Kreuzkorrelation mit vorher aufgenommenen Pfeifen [Röf+16, S. 86-88]. Die Kreuzkorrelation wird dabei über Multiplikation im Frequenzbereich berechnet. Wenn der Anteil der Kreuzkorrelation an der Autokorrelation einer Referenzpfeife einen Schwellwert überschreitet, ist die Pfeife erkannt. Zusätzlich muss noch eine Mindestamplitude im Zeitbereich überschritten werden.

Poore u. a. [Poo+15] vom Team RoboCanes der University of Miami vergleichen zwei Methoden. Die erste vorgeschlagene Methode ist sehr ähnlich zu der Methode der Bembelbots, d. h. sie überprüft, ob die Frequenz mit der höchsten Amplitude in einem bestimmten Bereich liegt. Eine Überprüfung der Länge des Geräuschs findet hier nicht statt. Die andere vorgeschlagene Methode ist, durch logistische Regression über das logarithmierte Energiespektrum ein Modell der Pfeife zu lernen.

Kapitel 3

Implementierung

Um die verschiedenen vorgestellten Methoden auf einem standardisierten Datensatz auszuwerten, wurde das Programm WhistleLab entwickelt. Es hätte auch die Möglichkeit gegeben, bereits vorhandene Audiosoftware zu verwenden und diese durch Plugins zu erweitern. Dies hätte allerdings die Einarbeitung in ein vorhandenes Softwaresystem erfordert. Es bestünde außerdem die Gefahr, dass manche benötigte Funktionalität damit gar nicht realisierbar ist. Daher wurde dieser Option nicht weiter nachgegangen.

Auf eine Implementierung im Framework der HULKs wurde in dieser Arbeit verzichtet, da die Lauffähigkeit auf dem Roboter zur Auswertung nicht nötig ist und keine weiteren Vorteile mit sich bringt.

3.1 WhistleLab

Die Software WhistleLab wurde unter Nutzung des Qt-Frameworks [Com17] mit einer grafischen Benutzeroberfläche in `C++` implementiert. Sie ist in zwei Komponenten aufgeteilt; die `Engine` und die `UI`. Diese laufen in separaten Threads, damit z. B. während der Auswertung eines Detektors die Benutzeroberfläche weiterhin reagiert. Sie kommunizieren über das Signal-Slot-System von Qt.

Die `UI` enthält die Klassen zur Darstellung der Benutzeroberfläche und Entgegennahme von Benutzereingaben. Über die Hauptmenüleiste kann eine Audiodatenbank geladen werden sowie das Evaluieren und Trainieren eines Detektors veranlasst werden. Im linken Unterfenster wird die Baumstruktur der Datenbank angezeigt und es kann ein Kanal zum Abspielen ausgewählt werden. Über das rechte Unterfenster kann das Abspielen des ausgewählten Kanals gesteuert werden.

Das zentrale Datenobjekt in der `Engine` ist die `SampleDatabase`. Diese ist hierarchisch aufgebaut und enthält auf der obersten Ebene eine Liste von Objekten der Klasse `AudioFile`. Ein `AudioFile` enthält wiederum eine Liste von `AudioChannel`-Objekten. Jeder `AudioChannel` beinhaltet das Audiosignal selbst als Folge von Samp-

les und eine Liste von Bereichen (definiert durch Start- und Endposition), in denen ein Pfiff enthalten ist.

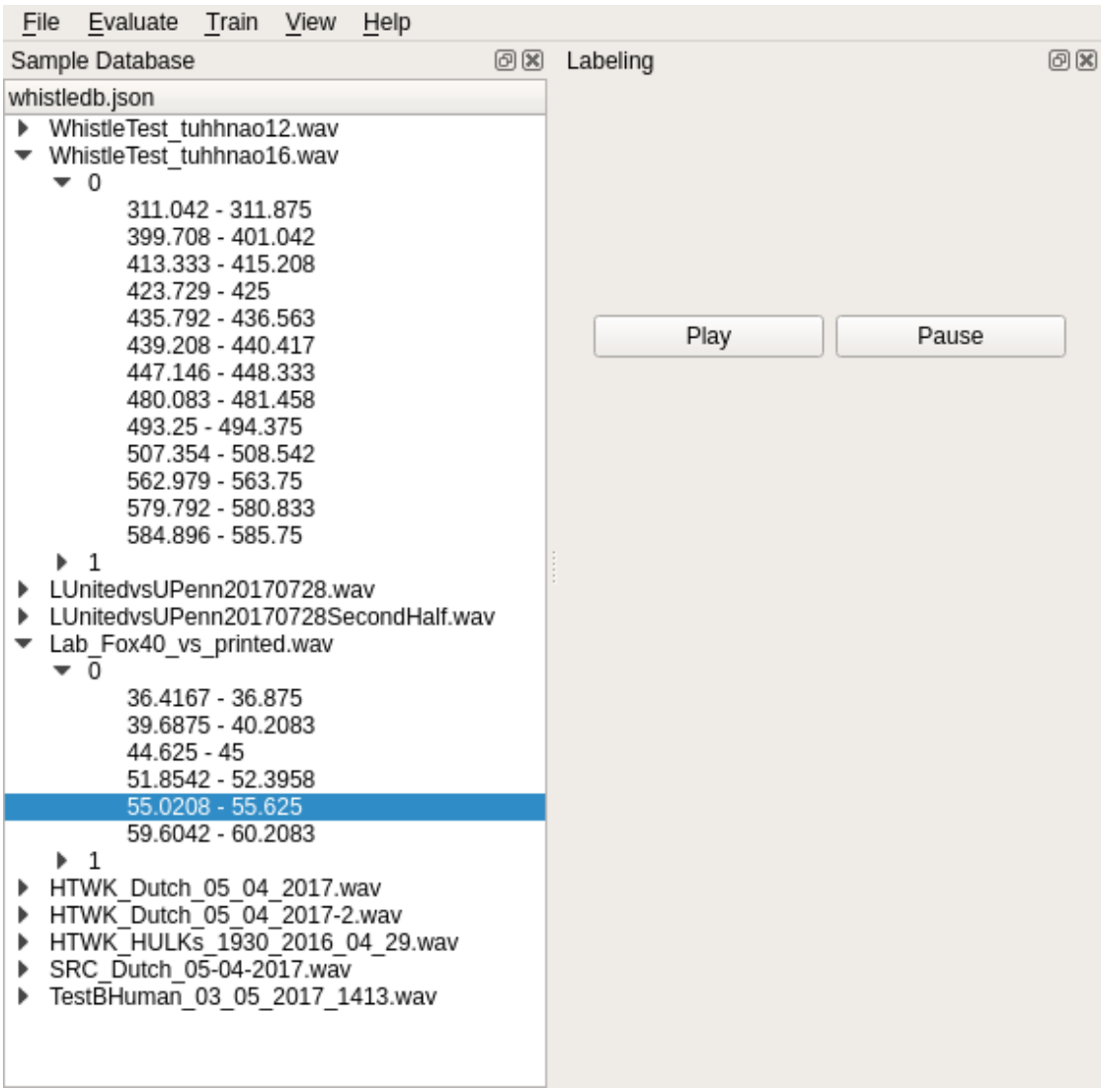

Abbildung 3.1: *Hauptfenster von WhistleLab. Links sind die annotierten Audiodateien und -kanäle gelistet. Rechts kann ein ausgewählter Audiokanal abgespielt werden.*

Für die Serialisierung der `SampleDatabase` wird die JavaScript Object Notation (JSON) verwendet. Dabei handelt es sich um ein textbasiertes Datenformat, das strukturierte Daten repräsentieren kann. Es wurde ausgewählt, da zum einen die Datenbank mit einem Texteditor erstellt werden musste, zum anderen Qt für JSON-(De)serialisierung bereits Klassen enthält. Die Audiodateien selbst werden in der JSON-Datei nur mit ihrem Namen referenziert. Bei der Deserialisierung werden sie dann separat geöffnet und

eingelesen. Dazu wird die Bibliothek libsndfile [CL] verwendet, die diverse Audiodateiformate unterstützt.

Alle Detektoren erben von der Basisklasse `WhistleDetectorBase`. Diese verwendet ein Pattern, das die automatische Registrierung der Detektoren in einer Liste ermöglicht, sodass bei einem neuen Detektor nicht diverse Stellen in der `UI` geändert werden müssen. Jeder Detektor muss die Methode `evaluate` implementieren, die den Algorithmus auf einer einzigen Audiodatei ausführt. Dazu wird `evaluate` eine Referenz auf ein Objekt der Hilfsklasse `EvaluationHandle` übergeben. Aus diesem kann der Detektor Samples lesen und darin die Zeitpunkte detektierter Pfeifen relativ zur aktuellen Leseposition eintragen. Die Methode `evaluateOnDatabase` ist in der `WhistleDetectorBase` implementiert und benutzt das `evaluate` der abgeleiteten Klasse, um automatisch von der gesamten `SampleDatabase` die in Kapitel 4 definierten Kennzahlen zu bestimmen.

Die Ergebnisse der Evaluation werden in der Datenstruktur `EvaluationResults` gespeichert. Von dort aus können sie an die grafische Oberfläche weitergegeben werden, um dem Benutzer angezeigt zu werden. Derzeit werden sie nur auf der Standardausgabe ausgegeben.

Zum Training können Detektoren die Methode `trainOnDatabase` implementieren. Der eigene Detektor implementiert diese Methode so, dass er sich in einen Trainingszustand versetzt und dann `evaluateOnDatabase` auf sich selbst aufruft. Im Trainingszustand findet nur die Merkmalsberechnung statt, aber nicht die Klassifizierung. Aus dem `EvaluationHandle` kann dann auf die Annotation zugegriffen werden, um den Merkmalsvektor als positives oder negatives Trainingsbeispiel abzuspeichern. So wird ein annotierter Trainingsdatensatz erzeugt, ohne dass die Merkmalsberechnung doppelt implementiert werden muss.

Für die Berechnung der DFT verwenden alle Detektoren die freie Bibliothek FFTW [FJ05]. Diese realisiert eine Art der schnellen Fourier-Transformation, die sich zur Laufzeit an die Rechenhardware anpasst.

Zum Training und Auswerten des neuronalen Netzes für den eigenen Detektor wird die Bibliothek FANN [Nis] benutzt. Diese wird zwar seit Ende 2015 nicht mehr weiterentwickelt und es fehlt z. B. eine ReLU-Aktivierungsfunktion, hat aber im Gegensatz zu modernen und umfangreicheren Bibliotheken den Vorteil, dass sie ein leicht zu benutzendes `C`-Interface besitzt.

Kapitel 4

Evaluation und Diskussion

In WhistleLab wurden sowohl der eigene Detektor als auch die Algorithmen von UNSW Sydney, den Nao Devils Dortmund, den Bembelbots und den HULKs implementiert. Vom UNSW-Algorithmus wird die Version von 2015 — mit dem zusätzlichen Schwellwert über die Lautstärke in der letzten Sekunde — getrennt von der 2016er Version — ohne diesen Schwellwert — aufgeführt.

Zunächst werden die Herkunft und die Zusammensetzung des verwendeten Datensatzes dargestellt. Anschließend wird jedes in Kapitel 1.2 vorgestellte Kriterium in Form einer Maßzahl konkretisiert und die entsprechenden Ergebnisse der Detektoren gelistet. Zuletzt werden die Ergebnisse der einzelnen Methoden diskutiert und ein Fazit gezogen.

Die Parameterwerte, mit denen die Evaluation durchgeführt wurde, sind in Anhang A aufgelistet.

4.1 Der Datensatz

Die verwendeten Audiodaten wurden alle mit den Mikrofonen von NAO-Robotern aufgenommen. Sie wurden bei verschiedenen Gelegenheiten mit unterschiedlichen Robotern gesammelt. Da es bei den HULKs nicht ohne weiteres möglich war, mit einem spielenden Roboter Audiodaten aufzuzeichnen, musste zur Aufnahme von echten Spielen ein weiterer Roboter neben das Spielfeld gestellt werden.

Es gibt sowohl Audiodateien mit 44.1 kHz als auch 48 kHz Abtastrate. Die Amplitudenauflösung beträgt bei allen 8 bit, es sind also 256 verschiedene Stufen möglich. Alle Dateien enthalten zwei Audiokanäle, von denen jedoch alle Detektoren nur den ersten benutzen. Insgesamt sind gut 82 min Audiomaterial vorhanden; darin sind 51 Pfiffe enthalten.

Die Audiodateien wurden alle mit dem freien Programm Audacity durchgehört und der Anfang und das Ende von Pfeifen ausgewählt. Dadurch können die entsprechenden Positionen (in der Einheit Samples) abgelesen werden (s. Abb. 4.1) und in die JSON-

Dateiname	Abtastrate	Szenario
WhistleTest_tuhhnao12.wav	48 kHz	Test unterschiedlicher Pfeifen bei den German Open 2017
WhistleTest_tuhhnao16.wav	48 kHz	Test unterschiedlicher Pfeifen bei den German Open 2017
LUnitedvsUPenn20170728.wav	48 kHz	Spiel beim RoboCup 2017
LUnitedvsUPenn20170728Half2.wav	48 kHz	Spiel beim RoboCup 2017
Lab_Fox40_vs_printed.wav	48 kHz	Laborversuch ohne Lärm mit zwei unterschiedlichen Pfeifen
HTWK_Dutch_05_04_2017.wav	44.1 kHz	Spiel bei den IranOpen 2017
HTWK_Dutch_05_04_2017-2.wav	44.1 kHz	Spiel bei den IranOpen 2017
HTWK_HULKs_1930_2016_04_29.wav	48 kHz	Testspiel bei einer Veranstaltung in Leipzig
SRC_Dutch_05-04-2017.wav	44.1 kHz	Spiel bei den IranOpen 2017
TestBHuman_03_05_2017_1413.wav	48 kHz	Test bei den German Open 2017

Tabelle 4.1: *Die verwendeten Audiodateien*

Datenbank eingetragen werden. Es wurde darauf geachtet, dass möglichst keine Pfeifen von anderen Spielfeldern als Pfeife gespeichert werden.

Die Aufnahmen vom Pfeifentest bei den German Open 2017 enthalten mehrere verschiedene Pfeifenmodelle. Es ist jedoch davon auszugehen, dass die Schiedsrichter sich dabei besonders angestrengt haben, sodass nicht alle Pfiffe in der Realität so kräftig sind. Beim Laborversuch sind Pfiffe von einer Fox-40-Pfeife und einer 3D-gedruckten Pfeife enthalten. Die bei den IranOpen 2017 aufgenommenen Dateien zeichnen sich dadurch aus, dass sie sehr viele Pfiffe von anderen Spielfeldern enthalten. Außerdem gab es dort häufig laute Ansagen. Beim Testspiel in Leipzig stand der aufnehmende Roboter sehr nah am Publikum, sodass deren Rufe deutlich lauter zu hören sind als die Pfiffe.

Eine Beobachtung beim Ansehen der Audiodateien ist, dass viele Pfiffe das Mikrofon übersteuern und somit zu rechteckförmigen Signalen werden. Dadurch entstehen auch Frequenzanteile über 12 kHz, obwohl die Mikrofone diesen Bereich gar nicht mehr erfassen.

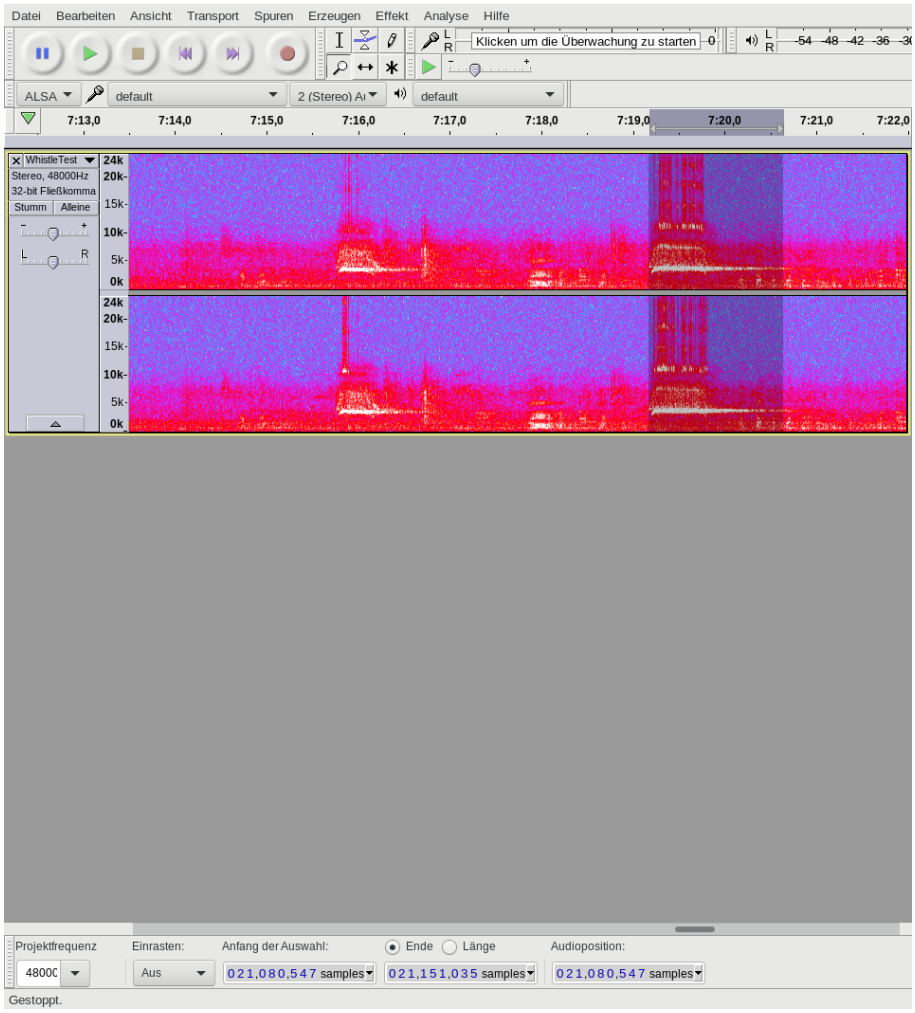

Abbildung 4.1: *Die Verwendung von Audacity zum Annotieren der Audiodaten. Anfang und Ende der Auswahl mussten anschließend manuell in eine JSON-Datei eingetragen werden.*

4.2 Trefferquote

Die Trefferquote ist der Quotient aus der Anzahl von detektierten wahren Pfeifen (*True Positives*) und wahren Pfeifen (*Positives*). Dabei spielt es für die Zählung der *True Positives* keine Rolle, wie viele Meldungen währenddessen gemacht werden, sondern nur, ob mindestens eine Meldung getroffen hat. Die Ergebnisse der Auswertung der Trefferquote befinden sich in Tabelle 4.2.

Detektor	Trefferquote	Prozentsatz
Bembelbots	42/51	82.4 %
HULKs	36/51	70.6 %
Nao Devils Dortmund	47/51	92.2 %
UNSW Sydney 2015	46/51	90.2 %
UNSW Sydney 2016	50/51	98.0 %
eigene Methode	50/51	98.0 %

Tabelle 4.2: *Die Trefferquoten der Detektoren*

4.3 Fehldetektionshäufigkeit

Zur neutralen Bewertung der Fehldetektionshäufigkeit sollte nicht einfach die Anzahl der Falschmeldungen gezählt werden. Bei einem Geräusch von z. B. 1 s Länge, das leicht mit einer Pfeife zu verwechseln ist, können verschiedene Detektoren bei der Anzahl der Falschmeldungen weit auseinander liegen. Ein Detektor beispielsweise, der eine kleine DFT-Größe benutzt, meldet bei diesem Geräusch deutlich mehr Fehldetektionen als einer, der nur wenige Spektren pro Sekunde verarbeitet. In der realen Auswirkung unterscheiden die beiden sich aber nicht, denn der Roboter würde bei diesem Geräusch in jedem Fall loslaufen. Um dies auszugleichen, werden Falschmeldungen nur dann gezählt, wenn der Detektor nicht in der letzten Sekunde schon eine Fehldetektion gemeldet hat. Die Ergebnisse der Auswertung der Fehldetektionshäufigkeit befinden sich in Tabelle 4.3.

Detektor	Anzahl der Fehldetektionen
Bembelbots	29
HULKs	27
Nao Devils Dortmund	104
UNSW Sydney 2015	8
UNSW Sydney 2016	26
eigene Methode	15

Tabelle 4.3: *Die Fehldetektionshäufigkeiten der Detektoren*

4.4 Kalibrierungsaufwand

Der Kalibrierungsaufwand eines Detektors lässt sich sehr grob durch die Anzahl der Parameter bewerten. Allerdings wird dabei nicht berücksichtigt, dass sich die Parameter

in ihrem Einfluss auf die Detektionsqualität sehr unterscheiden. In diesem Kontext sind die Daten in Tabelle 4.4 zu sehen.

Detektor	**Anzahl der Parameter**
Bembelbots	5
HULKs	4
Nao Devils Dortmund	12
UNSW Sydney 2015	9
UNSW Sydney 2016	8
eigene Methode	5

Tabelle 4.4: *Die Anzahlen der Parameter der Detektoren*

Die Kalibrierungsfreiheit wurde zusätzlich getestet, indem während der Auswertung alle Parameter über alle Dateien hinweg konstant gelassen wurden.

4.5 Rechenaufwand

Die praktisch relevante Messgröße im Zusammenhang mit dem Rechenaufwand ist die Laufzeit auf dem NAO-Roboter, die ein Detektor pro Zyklus des Softwaresystems benötigt. Da die Auswertung nicht auf dem NAO durchgeführt wurde, konnten diese Zeiten nicht ermittelt werden. Stattdessen wurde auf dem Rechner des Autors nicht die Laufzeit selbst gemessen, sondern das Verhältnis von der Rechenzeit zu der verarbeiteten Dauer von Audiodaten. Dazu misst WhistleLab die Thread-Zeit, die ein Detektor zwischen zwei Lesevorgängen verbringt, und dividiert diese durch die Dauer der gelesenen Daten. So ergibt sich theoretisch die Zeit, die der Algorithmus zum Verarbeiten von 1 s Audiodaten benötigen würde. Diese relative Laufzeit ist allerdings immer noch von der verwendeten Rechenhardware abhängig, sodass sich damit nicht entscheiden lässt, ob eine Methode auf dem NAO einsetzbar ist. Zur Bewertung der Methoden an sich ist die Laufzeit ohnehin nur begrenzt geeignet, da sie eine Eigenschaft der konkreten Implementierung und nicht der Methode selbst ist. Die Ergebnisse der Laufzeitauswertung befinden sich in Tabelle 4.5.

4.6 Online-Fähigkeit

Die Online-Fähigkeit wird anhand der sog. Reaktionszeit gemessen. Die Reaktionszeit ist die Zeit, die von dem annotierten Anfang der Pfeife bis zum ersten Zeitpunkt, an dem der Detektor die Pfeife meldet, vergeht. Dieser Zeitpunkt wird durch die Leseposition des Detektors in der Audiodatei definiert, d. h. er gibt an, wie viele Daten dem Detektor

Detektor	minimale Laufzeit	mittlere Laufzeit	maximale Laufzeit
Bembelbots	$373 \cdot 10^{-6}$	$430 \cdot 10^{-6}$	$4328 \cdot 10^{-6}$
HULKs	$157 \cdot 10^{-6}$	$188 \cdot 10^{-6}$	$831 \cdot 10^{-6}$
Nao Devils Dortmund	$776 \cdot 10^{-6}$	$899 \cdot 10^{-6}$	$6161 \cdot 10^{-6}$
UNSW Sydney 2015	$206 \cdot 10^{-6}$	$340 \cdot 10^{-6}$	$2879 \cdot 10^{-6}$
UNSW Sydney 2016	$202 \cdot 10^{-6}$	$239 \cdot 10^{-6}$	$2515 \cdot 10^{-6}$
eigene Methode	$703 \cdot 10^{-6}$	$848 \cdot 10^{-6}$	$6314 \cdot 10^{-6}$

Tabelle 4.5: *Die relativen Laufzeiten der Detektoren*

zur Verfügung standen. Die Reaktionszeit ist nur für *True Positives* ermittelbar. Alle Reaktionszeiten sind in Tabelle 4.6 aufgelistet.

Detektor	minimale Reaktionszeit	mittlere Reaktionszeit	maximale Reaktionszeit
Bembelbots	392 ms	438 ms	633 ms
HULKs	38 ms	516 ms	1387 ms
Nao Devils Dortmund	51 ms	83 ms	186 ms
UNSW Sydney 2015	214 ms	285 ms	655 ms
UNSW Sydney 2016	214 ms	282 ms	655 ms
eigene Methode	32 ms	91 ms	911 ms

Tabelle 4.6: *Die Reaktionszeiten der Detektoren*

4.7 Diskussion

Mit den Ergebnissen der quantitativen Auswertung können die Methoden nun bewertet werden.

4.7.1 Bembelbots

Der Detektor der Bembelbots hat die offensichtliche Schwäche, dass er eine Pfeife nicht mehr erkennt, wenn zusätzlich laute Geräusche im niedrigeren Frequenzbereich sind, wie auch in [TCP03, S. 3] beschrieben ist. Dies ist u. a. für das Nichterkennen der Pfiffe im Test bei den German Open 2017 verantwortlich, da der Lüfter des Roboters dort ein lautes Geräusch zwischen 500 Hz und 1 kHz verursachte. Außerdem wird die 3D-gedruckte Pfeife aus dem Laborversuch nicht erkannt, da ihre Spitzenfrequenz unter dem eingestellten Schwellwert liegt.

Die Fehldetektionen umfassen u. a. den Klang einer Kreissäge, die während des Pfeifentests bei den German Open 2017 beim Aufbau des benachbarten RoboCupRescue-Geländes benutzt wurde. 12 Fehldetektionen sind Pfiffe auf anderen Feldern.

Die Mindestpfeifendauer von 400 ms sorgt für eine entsprechende Reaktionszeit.

4.7.2 HULKs

Die Schwäche dieses Detektors liegt vor allem darin, dass bei seiner Entwicklung wenig Zeit zur Verfügung stand und dementsprechend die Eigenschaften der Pfeife hauptsächlich geraten wurden. Es war z. B. nicht bekannt, dass die Pfiffe häufig verzerrt sind und sich dadurch viel Leistung im Band oberhalb der Grundfrequenz der Pfeife befindet. Dies sorgt dafür, dass meistens nur der Nachhall der Pfeife erkannt wird. So lässt sich auch die hohe Reaktionszeit erklären.

Fehldetektionen finden hauptsächlich in Sprache statt, allerdings sind auch bei diesem Detektor 18 Pfiffe von anderen Feldern dabei.

Lediglich der niedrige Rechenaufwand und die geringe Anzahl an Parametern sind positiv bei diesem Detektor.

4.7.3 Nao Devils Dortmund

Die Trefferquote des Nao-Devils-Detektors zählt zu den besten. Die 4 nicht erkannten Pfiffe sind so leise, dass sie die entsprechenden Amplitudenschwellwerte nicht überschreiten.

Bei dieser Methode zeigt sich besonders die Blockierung von Fehldetektionen innerhalb von 1 s einer vorangegangenen Fehldetektion. Dies liegt an der hohen Anzahl untersuchter Spektren pro Sekunde, da eine geringe DFT-Größe und überlappende Fenster verwendet werden. Die Anzahl der Fehldetektionen ohne Kompensation beträgt 1406.

Trotzdem ist selbst die bereinigte Fehldetektionshäufigkeit immer noch 104. Die methodische Schwäche des Nao-Devils-Algorithmus liegt darin, dass nicht überprüft wird, ob die Amplitude zwischen den Obertönen niedriger wird. Das Überschreiten bestimmter Schwellwerte in bestimmten Frequenzbereichen trifft jedoch ebenso auf diverse andere Signale, z. B. sogar weißes Rauschen, zu.

Diese negative Eigenschaft kann auch nicht durch die gute Trefferquote und Reaktionszeit ausgeglichen werden. Die vergleichsweise hohe Laufzeit liegt vermutlich daran, dass alle Audiodaten durch die überlappenden DFT-Fenster zweimal verarbeitet werden.

4.7.4 UNSW Sydney

Der einzige nicht erkannte Pfiff des 2016er Detektors ist ein Test mit der 3D-gedruckten Trillerpfeife des Laborversuchs. Diese Pfeife zeichnet sich durch ihre besonders niedrige Grundfrequenz aus. Die vier weiteren nicht erkannten Pfiffe in der 2015er Version sind einfach nur zu leise gegenüber der Umgebung, sodass sie den über die letzte Sekunde berechneten Schwellwert nicht überschreiten.

Alle gemessenen Fehldetektionen der UNSW-Methode, sowohl in der Version von 2015 als auch von 2016, sind Pfeifen auf anderen Spielfeldern. Dies ist im Vergleich zu allen anderen betrachteten Detektoren ein positives Alleinstellungsmerkmal. Es zeigt, dass dieser Detektor ggf. in Kombination mit einer Lokalisierung der Geräuschquelle unschlagbar gut sein kann.

Die Reaktionszeit lässt sich offensichtlich darauf zurückführen, dass die Detektionsmethode eine Mindestdauer von 250 ms akzeptierten Spektren fordert, bevor eine Pfeife gemeldet wird.

Technisch gesehen zeichnet sich diese Methode dadurch aus, dass es kaum voreingestellte Schwellwerte gibt, sondern diese adaptiv berechnet werden.

4.7.5 Eigene Methode

Die eigene Methode hat zusammen mit dem UNSW-2016-Algorithmus die höchste Trefferquote. Der einzige nicht erkannte Pfiff kommt vom Pfeifentest bei den German Open 2017. Derselbe Pfiff in der mit einem anderen Roboter aufgenommenen Datei wurde jedoch erkannt.

8 der 15 Fehldetektionen sind Pfiffe auf anderen Feldern. Die restlichen 7 sind lautes Publikum direkt neben dem Roboter bzw. Ansagen in der Halle. Es ist möglich, dass diese Geräuschklassen bei der zufälligen Auswahl der Trainingsbeispiele unterrepräsentiert sind.

Die Gewichte des neuronalen Netzes sind in die Anzahl der Parameter nicht einbezogen. Es ist aber erhofft, dass das neuronale Netz nicht für andere Pfeifen oder Umgebungsbedingungen neu trainiert werden muss.

Aufgrund der nicht vorhandenen Mindestdauer eines Pfiffs bei dieser Methode gehört die Reaktionszeit zu den besten. Die Laufzeit ist jedoch eine der höchsten, da sowohl die Merkmale berechnet werden müssen als auch der Klassifizierer ausgeführt werden muss.

4.8 Fazit

Betrachtet man nur die Trefferquote und die Fehldetektionshäufigkeit, stellt man fest, dass die Bembelbots-Methode und die HULKs-Methode durch beide UNSW-Methoden und die eigene Methode dominiert werden. Gegen den Einsatz der Nao-Devils-Methode spricht die hohe Zahl an Fehldetektionen. In den anderen Kriterien unterscheiden sich die Detektoren nicht so viel, dass es Defizite in den ersten zwei Kriterien wettmachen könnte. Die Reaktionszeiten sind alle in einer Größenordnung, in der sie gegenüber der Laufgeschwindigkeit nicht von Bedeutung sind.

Damit bleiben die UNSW-Methoden und die eigene Methode als Kandidaten für die beste Lösung. Für den UNSW-Algorithmus spricht die Tatsache, dass dort wirklich nur Pfiffe erkannt werden, auch wenn sie manchmal von anderen Spielfeldern kommen. Außerdem ist der gesamte Algorithmus nachvollziehbar und enthält keine „Black-Box“ wie ein neuronales Netz. Die eigene Methode ist von den Kennzahlen her ein Kompromiss zwischen der 2015er und 2016er Version von UNSW.

Kapitel 5

Zusammenfassung und Ausblick

In dieser Arbeit wurden unterschiedliche Methoden zur Detektion von Pfeifengeräuschen zur Benutzung auf dem NAO-Roboter vorgestellt, implementiert und verglichen. Damit wurde das in Kapitel 1.2 gesetzte Ziel erreicht.

Je nach Gewichtung der Kriterien zählen der Detektor von UNSW Sydney und der selbst entwickelte Detektor zu den besten. Es muss dabei zwischen Trefferquote und Fehldetektionshäufigkeit abgewogen werden. Für beide Versionen des Detektors von UNSW Sydney spricht die Abwesenheit von Fehldetektionen, abgesehen von Pfeifen auf anderen Feldern. Der selbst entwickelte Detektor ist bezüglich der Trefferquote ebenso gut wie der 2016er UNSW-Detektor. Er hat aber nicht nur bei Pfeifen auf anderen Feldern Fehldetektionen, sondern auch z. B. in besonders lauten Rufen von Zuschauern direkt neben dem Roboter, dafür aber insgesamt weniger davon. Rechenaufwand und Online-Fähigkeit sind bei allen Detektoren im zulässigen Rahmen.

Trotzdem gibt es noch Bereiche, in denen mit weiteren Bemühungen eine Verbesserung erzielt werden kann. Dazu zählen sowohl die selbst entwickelte Pfeifendetektionsmethode als auch das zur Auswertung und zum Training verwendete Programm WhistleLab. Außerdem steht als nächster Schritt die Implementierung ausgewählter Methoden in einem Softwaresystem an, das tatsächlich auf dem Roboter läuft.

5.1 Detektionsmethode

Die in dieser Arbeit entwickelte Lösung zur Pfeifenerkennung bietet zahlreiche Ansätze für weitere Entwicklungen. So gibt es noch Potenzial bei der Auswahl der Merkmale. Eine Möglichkeit ist, eine Abwandlung von Mel-Frequenz-Cepstral-Koeffizienten, wie in [Poo+15] vorgeschlagen wird, zu verwenden. Während die Mel-Skala eine besonders hohe Auflösung im Bereich bis 1 kHz hat, in dem Pfeifen normalerweise keine Frequenzanteile haben, könnte eine andere Skala gewählt werden, die sich z. B. an die

geschätzte Grundfrequenz der Pfeife anpasst. Es sind auch Merkmale denkbar, die die Phaseninformation aus der DFT verwenden.

Der Klassifizierungsansatz mit dem künstlichen neuronalen Netz kann noch verbessert werden. Insbesondere sind — wie bei Maschinenlernverfahren üblich — bessere Ergebnisse zu erwarten, wenn mehr Daten vorhanden sind. Außerdem können hier die Hyperparameter des neuronalen Netzes, d. h. die Aktivierungsfunktionen und die Anzahl der Neuronen in den verdeckten Schichten, optimiert werden.

Bisher verwendet der Detektor nur einen Audiokanal. Durch die Benutzung mehrerer Mikrofone des NAOs kann vermutlich mehr Robustheit erreicht werden, z. B. indem eine Pfeife nur gemeldet wird, wenn sie auf mehreren Kanälen detektiert wurde.

Es können auch einige Ideen anderer Detektoren integriert werden. Dazu gehören überlappende DFT-Fenster, damit trotz der Hann-Fensterung alle Werte in ein Spektrum eingehen. Außerdem könnte die Forderung einer Mindestdauer, wie sie u. a. bei UNSW Sydney erfolgreich umgesetzt wird, problemlos hinzugefügt werden.

5.2 Weiterentwicklung von WhistleLab

Zugunsten einer im Zeitrahmen dieser Arbeit ausreichend nutzbaren Software wurden zahlreiche denkbare Features in WhistleLab nicht implementiert. WhistleLab sollte ursprünglich auch das Annotieren von Audiodaten ermöglichen, was durch die Verwendung von Audacity umgangen wurde. Diese Funktionalität wäre aber immer noch komfortabler als das manuelle Eintragen der abgelesenen Bereiche in die JSON-Datei. Die Annotation erfolgt dann idealerweise gleich in mehrere Negativklassen, z. B. Pfeifen auf anderen Feldern und andere laute Geräusche, damit diese beim Erstellen eines Trainingsdatensatzes speziell behandelt werden können.

Die von den Detektoren gemeldeten Pfeifen könnten grafisch unter einer Visualisierung der Audiospur dargestellt werden, statt wie bisher nur in Textform ausgegeben zu werden. Somit könnte man auf einen Blick sehen, an welchen Stellen Fehldetektionen stattfinden und welche Eigenschaften des Signals dafür verantwortlich sein könnten. Dies wäre hilfreich dabei, systematisch gegen Fehldetektionen vorzugehen.

Literatur

[Bem16] Bembelbots. *Bembelbots 2016 Whistle Detector*. 30. Nov. 2016. URL: https://github.com/Bembelbots/NaoWhistleDetection (besucht am 30. 09. 2017).

[CL] Erik de Castro Lopo. *libsndfile*. URL: http://www.mega-nerd.com/libsndfile (besucht am 01. 10. 2017).

[Com14] RoboCup Techical Committee. *RoboCup Standard Platform League (NAO) Technical Challenges*. 19. Juni 2014. URL: http://spl.robocup.org/wp-content/uploads/downloads/Challenges2014.pdf (besucht am 30. 09. 2017).

[Com15] RoboCup Technical Committee. *RoboCup Standard Platform League (NAO) Rule Book*. 1. Juli 2015. URL: http://spl.robocup.org/wp-content/uploads/downloads/Rules2015.pdf (besucht am 30. 09. 2017).

[Com16] RoboCup Technical Committee. *RoboCup Standard Platform League (NAO) Rule Book*. 24. Juni 2016. URL: http://spl.robocup.org/wp-content/uploads/downloads/Rules2016.pdf (besucht am 30. 09. 2017).

[Com17] The Qt Company. *Qt*. 2017. URL: https://www.qt.io (besucht am 01. 10. 2017).

[Dor16] Nao Devils Dortmund. *Nao Devils Dortmund 2016 Whistle Detector*. 8. Nov. 2016. URL: https://github.com/NaoDevils/CodeRelease2016/tree/master/Src/Modules/Modeling/WhistleDetector (besucht am 30. 09. 2017).

[FJ05] Matteo Frigo und Steven G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005), S. 216–231.

[Hal+15] Brad Hall u. a. "RoboCup SPL 2015 Champion Team Paper". In: *RoboCup 2015: Robot World Cup XIX*. Hrsg. von Luis Almeida u. a. Bd. 9513. Lecture Notes in Artificial Intelligence. Springer, 2015, S. 72–82.

[Ham14] Thomas Hamböck. *Austrian Kangaroos 2014 Whistle Detector*. 2014. URL: http://www.austrian-kangaroos.com/public/code/WhistleDetector.tgz (besucht am 30. 09. 2017).

[Har78] Frederic J. Harris. “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform”. In: *Proceedings of the IEEE* 66.1 (1978), S. 51–83.

[HS15] Thomas Hamböck und Dietmar Schreiner. *Austrian Kangaroos Team Research Report 2014*. Jan. 2015. URL: http://www.austrian-kangaroos.com/public/papers/AustrianKangaroos_ResearchReport_2014.pdf (besucht am 30. 09. 2017).

[KA00] Hiroaki Kitano und Minoru Asada. “The RoboCup humanoid challenge as the millenium challenge for advanced robotics”. In: *Advanced Robotics* 13.8 (2000), S. 723–736.

[Kit98] Hiroaki Kitano, Hrsg. *RoboCup-97: Robot Soccer World Cup I*. Bd. 1395. Lecture Notes in Artificial Intelligence. Springer, 1998.

[Nis] Steffen Nissen. *FANN*. URL: http://leenissen.dk/fann/wp (besucht am 01. 10. 2017).

[PLJ15] Fernando Puente León und Holger Jäkel. *Signale und Systeme*. 6. Aufl. De Gruyter, 2015.

[Poo+15] Kyle Poore u. a. “Single- and Multi-Channel Whistle Recognition with NAO Robots”. In: *RoboCup 2014: Robot World Cup XVIII*. Hrsg. von Reinaldo A. C. Bianchi u. a. Bd. 8992. Lecture Notes in Artificial Intelligence. Springer, 2015, S. 245–257.

[Rie+16] Nicolas Riebesel u. a. *Team Research Report 2016*. 1. Dez. 2016. URL: https://www.hulks.de/_files/TRR_2016.pdf (besucht am 30. 09. 2017).

[Rob17] SoftBank Robotics. *NAO - Technical overview*. 2017. URL: http://doc.aldebaran.com/2-1/family/robots/index_robots.html (besucht am 29. 09. 2017).

[Roj96] Raúl Rojas. *Neural Networks. A Systematic Introduction*. Springer, 1996.

[Röf+16] Thomas Röfer u. a. *B-Human Team Report and Code Release 2016*. 1. Dez. 2016. URL: https://www.b-human.de/downloads/publications/2016/coderelease2016.pdf (besucht am 30. 09. 2017).

[Syd15] UNSW Sydney. *UNSW Sydney 2015 Whistle Detector*. 4. Okt. 2015. URL: https://github.com/UNSWComputing/rUNSWift-2015-release/blob/master/image/home/nao/data/behaviours/audio/whistle_detector.py (besucht am 30. 09. 2017).

[Syd16] UNSW Sydney. *UNSW Sydney 2016 Whistle Detector*. 9. Okt. 2016. URL: https://github.com/UNSWComputing/rUNSWift-2016-release/blob/master/image/home/nao/whistle/whistle_detector.py (besucht am 30. 09. 2017).

[TCP03] Dian Tjondronegoro, Yi-Ping Phoebe Chen und Binh Pham. “Sports Video Summarization using Highlights and Play-Breaks”. In: *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval*. MIR ’03. ACM, 2003, S. 201–208.

Anhang A

Verwendete Parameterwerte

Bei allen Detektoren anderer SPL-Teams wurden, sofern möglich, die Parameter verwendet, die in deren Veröffentlichungen enthalten waren. Für die eigene Methode sind die Gewichte des neuronalen Netzes nicht aufgeführt.

Parameter	**Wert**
Puffergröße der DFT	50 ms
Mindestfrequenz f_{min}	2 kHz
Mindestamplitude a_{min}	-20 dB
Mindestdauer t_{min}	400 ms
Filterstärke n_{smooth}	3

Tabelle A.1: *Die verwendeten Parameterwerte für die Bembelbots-Methode*

Parameter	**Wert**
Puffergröße der DFT	8192
Anfang des Pfeifenbandes f_{min}	2 kHz
Ende des Pfeifenbandes f_{max}	4 kHz
Schwellwert $c_{\text{threshold}}$	50

Tabelle A.2: *Die verwendeten Parameterwerte für die HULKs-Methode*

Parameter	**Wert**
Puffergröße der DFT	1024
Anfang des Pfeifenbandes f_{min}	2 kHz
Ende des Pfeifenbandes f_{max}	4 kHz
Mindestamplitude a_0	35
Mindestfaktor 1. Oberton c_1	1.8
Höchstfaktor 1. Oberton c_2	2.2
Mindestamplitude 1. Oberton a_1	2
Mindestfaktor 2. Oberton c_3	2.8
Höchstfaktor 2. Oberton c_4	3.2
Mindestamplitude 2. Oberton a_2	2
Mindestanzahl akzeptierter Spektren n_1	4
Höchstanzahl nicht akzeptierter Spektren n_2	4

Tabelle A.3: *Die verwendeten Parameterwerte für die Nao-Devils-Methode*

Parameter	**Wert**
Puffergröße der DFT	1024
Anfang des Pfeifenbandes f_{min}	2 kHz
Ende des Pfeifenbandes f_{max}	4 kHz
Hintergrundfaktor $c_{\text{background}}$	0.7
Pfeifenfaktor c_{whistle}	2.5
Zeitfaktor c_{temporal}	5
Anzahl der Buckets n_{bucket}	10
Mindestdauer t_1	250 ms
Höchstdauer nicht akzeptierter Spektren t_2	83 ms

Tabelle A.4: *Die verwendeten Parameterwerte für die UNSW-Methode*

Parameter	**Wert**
Puffergröße der DFT	2048
Anfang des Pfeifenbandes f_{min}	2 kHz
Ende des Pfeifenbandes f_{max}	4 kHz
Minimalamplitude a_{min}	0.05
Amplitudenfaktor c_{minmax}	0.01

Tabelle A.5: *Die verwendeten Parameterwerte für die eigene Methode*

Anhang B

Inhalte der CD

Verzeichnis	**Inhalt**
/Bachelorarbeit	Kompilierte PDF und die zur Erstellung benötigten Quelldateien
/Audio	Die Audiodateien im WAV-Format inklusive Annotation im JSON-Format
/Software	Die Quelldateien der für diese Arbeit entwickelten Software
/Literatur	Einige der referenzierten Veröffentlichungen