

Automatisiertes Detektieren und Ausmessen von Waldschnepfenbalzrufen

Tobias Heeb

Peter Michael



Automatisiertes Detektieren und Ausmessen von Waldschnepfenbalzrufen

Tobias Heeb

Peter Michael

10. August 2016

eine Bachelorarbeit der Interstaatlichen Hochschule für Technik Buchs
in Zusammenarbeit mit der Schweizerischen Vogelwarte Sempach

Referent: Prof. Dr. Klaus Frick

Koreferent: Prof. Dr. Norbert Frei

Abstrakt

Ein Fokus der Schweizerischen Vogelwarte Sempach ist die Artenförderung und -überwachung der Waldschnepfe, einer heimlich lebenden und schwer zu beobachtenden Vogelart. Um genauere Daten über ihren aktuellen Bestand zu erhalten, werden Audioaufnahmen an verschiedenen Standorten gemacht. Diese Aufnahmen werden bisher manuell nach Balzrufen durchsucht; eine ermüdende und fehleranfällige Vorgehensweise.

Die vorliegende Arbeit beschäftigt sich mit der automatischen Detektion von Waldschnepfenbalzrufen in langen Audiosequenzen. Dazu wurden Detektionsalgorithmen basierend auf den mathematischen Signalverarbeitungskonzepten (Fourier- und Radontransformation, Spektrogrammanalyse etc.) erarbeitet. Diese erlauben es zusätzlich, die detektierten Balzrufe zu quantifizieren, was eine Grundvoraussetzung für die Individualerkennung darstellt.

Die Algorithmen wurden in einer eigenen Software realisiert. Importierte Audiosequenzen können automatisch nach Rufen durchsucht und die gefundenen Rufe ausgemessen werden. Sie können anschliessend visualisiert und exportiert werden.

Inhaltsverzeichnis

1	Dank	1
2	Einleitung	3
2.1	Ausgangslage	3
2.1.1	Die Vogelwarte Sempach	3
2.1.2	Die Waldschnepfe	3
2.1.3	Der Balzruf im Detail	3
2.2	Arbeitsweise der Vogelwarte	4
2.2.1	Auswertung der Aufnahmen	4
2.3	Beitrag dieser Arbeit	4
2.3.1	Themenschwerpunkte	4
2.3.2	Zielsetzung	5
2.3.3	Methodisches Vorgehen	6
2.4	BatScope – eine Zukunftsperspektive	6
3	Ruferkennung	7
3.1	Rufcharakterisierung	7
3.1.1	Variabilität des Pfuizlautes	8
3.1.2	Merkmale der Signalverarbeitung	11
3.2	Detektions- und Beurteilungsverfahren („Selektion“)	14
3.2.1	Einzelflankenerkennung	17
3.2.2	Gruppierung der Einzelflanken	18
3.2.3	Spektrogrammtests	20
3.2.4	Radon-Transformation	24
3.2.5	Verifizierung der verwendeten Features	27
3.3	Demonstration der Fähigkeiten – Benchmark	28
4	Software	40
4.1	Funktionen	40
4.2	Aufbau	40
4.3	Ordnerstruktur	40
4.3.1	Konfiguration - config	41
4.3.2	Logdatei - logs	41
4.3.3	Datenverzeichnis - projects	41
4.3.4	R-Skripte - scripts	41

4.3.5	Pfuizdatenbank - whistledb	42
4.3.6	Programmdatei - WoodcockScope.jar	42
4.4	Externe Java Bibliotheken	42
4.5	JavaFX	42
4.6	Lambda-Ausdrücke	43
4.7	Persistenz	43
4.8	Skriptsprache R	44
4.8.1	Verwendete Pakete	45
4.8.2	Anwendung	45
4.9	Java zu R Schnittstelle - Rserve	45
4.9.1	Java Client	45
4.9.2	voidEval vs. eval	47
4.9.3	Rserve-Serverinstanzen starten	48
4.9.4	Auslastung von mehreren Kernen	49
4.9.5	Effiziente Verarbeitung von Aufnahme-dateien	51
4.10	Benutzeroberfläche	52
4.10.1	Projekte	53
4.10.2	Rufanalyse	53
4.10.3	Analysedetails	54
4.11	Import	54
4.11.1	Metadaten	54
4.11.2	Automatischer Import aus SM3	54
4.11.3	Manueller Import	54
4.12	Export	54

5 Appendix A - theoretische Grundlagen 55

5.1	Transformationen im Allgemeinen	55
5.2	Diskrete Fourier-Transformation	56
5.3	Kreuzkorrelation	58
5.4	Spektrogramme	59
5.5	Fensterfunktionen	60
5.6	Kurzzeit-Fourier-Spektrum	63
5.7	Radon-Transformation	63

1 Dank

Damit diese Arbeit in der vorliegenden Qualität und mit dem vorliegenden Ergebnis geschrieben werden konnte, waren wir auf das Fachwissen und das Knowhow diverser Personen angewiesen. Diesen möchten wir an dieser Stelle unseren herzlichen Dank aussprechen. Ohne ihre Unterstützung, sowie ihren tatkräftigen Einsatz hätte diese Arbeit nicht in dieser Form realisiert werden können.

Der Referent Prof. Dr. Klaus Frick, sowie der Korreferent Prof. Dr. Norbert Frei haben uns während der gesamten Bearbeitungsphase dieser Bachelorarbeit unterstützt. In diversen Gesprächen konnten wir von ihrem grossen Wissen, ihren wertvollen Inputs, sowie von den konstruktiven Diskussionen profitieren. Sie haben uns mit viel Elan dabei unterstützt, die gesetzten Ziele zu erreichen und damit ein möglichst gutes Produkt für unseren Partner, die Vogelwarte Sempach, herzustellen. Standen wir vor Hürden, konnten wir in gemeinsamen Diskussionen mit ihnen neue Ideen und Perspektiven erarbeiten, die uns jeweils einen weiteren Schritt vorwärtsgebracht haben. Auch in den gemeinsamen Meetings mit der Vogelwarte und der WSL standen sie uns mit Rat und Tat zur Seite. Die Kompetenzen von Prof. Dr. Klaus Frick in Signal- und Bildverarbeitung, sowie die Erfahrungen von Prof. Dr. Norbert Frei in Softwaretechnik konnten wir in diese Arbeit einbringen und bereichernd nutzen. Das regelmässige Besprechen von unterschiedlichen Lösungsvarianten und die verständlichen Erklärungen zu einzelnen Themenpunkten haben uns sehr geholfen. Wir möchten uns deshalb herzlich bei Prof. Dr. Klaus Frick und bei Prof. Dr. Norbert Frei für ihre wertvollen Beiträge bedanken.

Ohne unseren Partner, die Vogelwarte Sempach, hätten wir dieses Thema nicht aufnehmen können. Thomas Sattler, Leiter der Abteilung «Überwachung der Vogelwelt», sowie Pierre Mollet, Mitarbeiter der Abteilung «Förderung der Vogelwelt» und Projektleiter des Artenförderungsprojektes zur Waldschnepfe, glaubten seit Beginn der Bachelorarbeit an die Idee und das Potenzial einer softwarebasierten Detektion von Waldschnepfenrufen. Sie ermöglichten uns durch die Partnerschaft eine vertiefte Auseinandersetzung mit diesem spannenden Thema. Durch ihr Vertrauen in die Idee und ihren grossen Einsatz konnten wir das uns fremde Gebiet der Artenförderung Waldschnepfe mit unserem Studium in Ingenieurinformatik verbinden. Thomas Sattler und Pierre Mollet unterstützten uns in diversen Gesprächen mit ihrem tiefen Wissen über die Waldschnepfe, sowie über die Prozesse in der Artenförderung. Ihr Fachwissen und ihre Erfahrungen lieferten wertvolle Einblicke in die Praxis und in das laufende Projekt. Sie bedienten uns mit wertvollem Material und waren Neuem gegenüber aufgeschlossen. So konnten wir uns

auf kreative und vielseitige Art einbringen. Sie befürworteten unsere Ideen und leisteten uns mit wertvollen Beiträgen eine grosse Unterstützung. Wir möchten uns herzlich bei der Vogelwarte und speziell bei Thomas Sattler und Pierre Mollet für die spannende Zusammenarbeit bedanken.

Eine weitere wertvolle Zusammenarbeit konnten wir mit der WSL, der Eidgenössischen Forschungsanstalt für Wald, Schnee und Landschaft, aufbauen. Dr. Martin Obrist und Dr. Ruedi Bösch standen uns mit ihrer praktischen Erfahrung in einigen Gesprächen zur Seite. Die Diskussionen mit ihnen halfen uns, die Software und die Algorithmen zu optimieren und weiterzuentwickeln. Durch den offenen Austausch über die Software Batscope konnten wir das bestehende Konzept der Software als Basis für die zu entwickelnde Detektionssoftware für Waldschnepfenrufe benutzen. Ihre Offenheit gegenüber dem erweiterten Einsatz von Batscope bietet der softwarebasierten Detektion von Waldschnepfenrufen auch nach dieser Bachelorarbeit eine Zukunft. Wir möchten uns bei Dr. Martin Obrist und Dr. Ruedi Bösch herzlich für ihren offenen Austausch und ihre wertvollen Beiträge bedanken.

Buchs, August 2016

2 Einleitung

Dieses Kapitel beschreibt die Ausgangslage, den fachwissenschaftlichen und anwendungsorientierten Kontext, die Zielsetzung und das methodische Vorgehen.

2.1 Ausgangslage

2.1.1 Die Vogelwarte Sempach

Die Schweizerische Vogelwarte Sempach ist eine gemeinnützige Stiftung. Als international anerkanntes Institut mit rund 100 Mitarbeitenden überwacht sie die einheimische Vogelwelt, geht den Ursachen der Bedrohung der Vogelwelt auf den Grund und setzt gemeinsam mit Partnern Schutz- und Fördermassnahmen um. Bei der landesweiten Überwachung der Vogelbestände kann sie auf die Unterstützung von über 2 000 freiwilligen Mitarbeiterinnen und Mitarbeitern zählen.

2.1.2 Die Waldschnepfe

Die Waldschnepfe ist eine heimlich lebende Waldvogelart, die in den letzten Jahrzehnten markante Bestandsrückgänge zu verzeichnen hatte. Etwas auffälliger sind nur die Männchen auf dem Balzflug im Frühling. Dabei geben sie einen Balzgesang von sich, der aus einer Rufreihe besteht, die zwischen höheren „Pfuiz-Lauten“ und einem tieferen „Quorren“ abwechselt. Es besteht die Hypothese, dass anhand dieser Rufreihen Individuen unterschieden werden können. Zusammen mit lokalen Ornithologen führt die Schweizerische Vogelwarte in den Waadtländer Voralpen seit 2013 ein Akustikprojekt durch. Dabei wurden bis 2015 an insgesamt 58 Aufnahmeabenden hunderte von Balzgesängen manuell aufgenommen. Typische Aufnahmelängen sind 120 min am Stück. Seit dem Frühling 2016 sind auch Aufnahmen aus Amden vorhanden.

2.1.3 Der Balzruf im Detail

Waldschnepfen sind nicht territorial, dies bedeutet, sie markieren ihr Revier nicht mit einem Ruf. Im Balzflug geben sie jedoch einen sehr charakteristischen Ruf von sich, welcher aus mehreren Quorren gefolgt von einem Pfuizen besteht (siehe Abbildung 1). Das Pfuizen befindet sich üblicherweise in einem Frequenzband zwischen 4 und 13 kHz und das Quorren im Bereich bis 1.5 kHz. Quorren ohne Pfuizen gibt es selten und kann nach Aussage der Vogelwarte vernachlässigt werden. Auch Pfuizen kann ohne Quorren

vorkommen. Dieser Fall kann auftreten, wenn die Distanz zwischen Aufnahmemikrofon und dem rufenden Vogel zu gross wird.

Da in dieser Arbeit das Quorren alleine nicht als Ruf identifiziert wird, kann das Pfuizen als Anker für die Ruferkennung verwendet werden. Der Balzruf der Waldschnepfe ist sehr charakteristisch und kann daher relativ einfach von anderen Geräuschen und Rufen unterschieden werden. Die Klassifizierung sollte deshalb zwei Gruppen umfassen: Waldschnepfenrufe und der Rest.

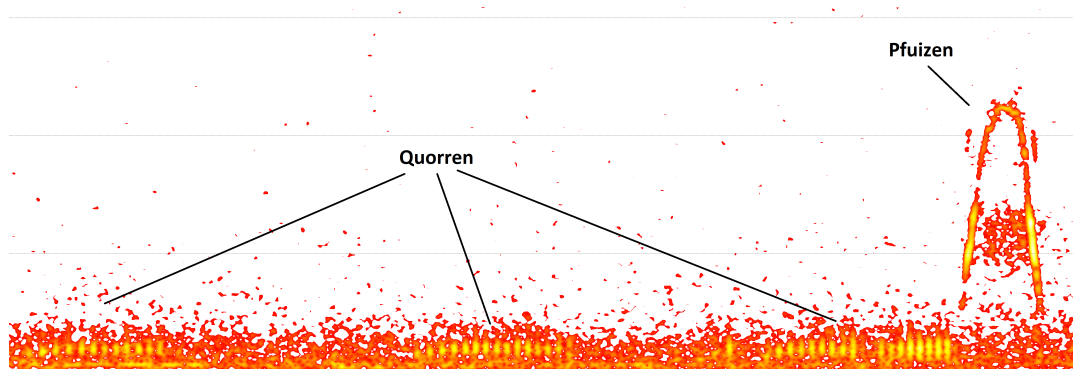


Abbildung 1: Spektrogramm des Balzrufs einer Waldschnepfe.

2.2 Arbeitsweise der Vogelwarte

2.2.1 Auswertung der Aufnahmen

Die Vogelwarte Sempach hat schon mehrere Aufnahmen durchgeführt und ausgewertet. Die Auswertung findet aktuell von Hand statt. Die Ausmessung von Hand ist jedoch eine zeitaufwändige und monotone Arbeit. Diese Arbeit soll so weit möglich automatisiert werden. Damit können grössere Datenmengen effizient ausgewertet werden und wichtige Personalressourcen für andere Aufgaben eingesetzt werden.

2.3 Beitrag dieser Arbeit

2.3.1 Themenschwerpunkte

Bei dieser Arbeit geht es um die Anwendung von Mustererkennung auf Vogelstimmen. Im Bereich der Mustererkennung hat der Stand der Forschung in den letzten Jahren grosse Fortschritte erzielt. Trend in diesem Gebiet ist aktuell der Forschungsbereich Deep Learning.

Wegen der spezifischen Thematik wurde für diese Aufgabenstellung jedoch eine andere Methode angewandt. Auch wäre die Vertiefung in das breite und komplexe Gebiet des Deep Learnings zeitlich nicht möglich gewesen. Der Vorteil von Deep Learning läge darin, dass die Erweiterung auf andere Arten einfacher ist, da bei den Deep Learning Verfahren kein a priori Wissen einer einzelnen Art verwendet wird [1]. In dieser Arbeit wird jedoch ein Verfahren verwendet, in dem sehr viel a priori Wissen über den Waldschnepfenbalzruf steckt. Es wäre deshalb nicht ohne weiteres möglich, das Verfahren auch auf andere Arten zu anzuwenden.

Diese Bachelorarbeit setzt sich aus zwei Themenschwerpunkten zusammen. Es wurden für die Erarbeitung der Algorithmen Erkenntnisse aus dem Bereich Signalverarbeitung verwendet. Der zweite Themenschwerpunkt ist die Softwareentwicklung. Folgende Fachgebiete wurden für diese Arbeit tangiert:

- Signalverarbeitung (FFT)
- Mustererkennung (Korrelation)
- Bildverarbeitung (Radon)
- Softwareentwicklung (JavaFX, Rserve)

2.3.2 Zielsetzung

Das erste Ziel dieser Bachelorarbeit ist, aus einer Vielzahl von unklassifizierten Akustiksequenzen die Balzrufe der Waldschnepfe automatisch zu extrahieren und die entsprechenden Koordinaten (Ort, Datum, Uhrzeit) in einer Datenbank abzulegen. Das zweite Ziel ist das automatische Berechnen von aussagekräftigen Parametern aus den detektierten Balzrufen (etwa nach Mulhauser und Zimmermann [2], oder Hoodless et al. [3]). Diese Parameter (Frequenz, Dauer etc.) werden ebenfalls in einer Datenbank abgelegt. Als Produkt wird eine schlanke und intuitive Software angestrebt, die von den Mitarbeitern der Vogelwarte Sempach einfach bedient werden kann.

Diese Ziele sind Grundvoraussetzung für die Bestimmung von nicht markierten, balzenden Waldschnepfenmännchen mit bioakustischen Methoden. Zusätzlich möchte die Vogelwarte dadurch das Potential von autonomen Aufnahmegeräten evaluieren.

Da in dieser Bachelorarbeit für die Softwareentwicklung RServe als Schnittstelle zwischen der Programmiersprache Java und der Skriptsprache R verwendet wird, kann als Nebenziel die Untersuchung der Tauglichkeit von RServe als Schnittstelle genannt werden.

2.3.3 Methodisches Vorgehen

In einem ersten Schritt wurde die Entwicklung der Algorithmen angestrebt. Danach wurden parallel die Software entwickelt und die Algorithmen optimiert.

Ziel 1 : Es muss ein geeignetes Verfahren gefunden werden, damit Waldschnepfenbalzrufe auf Aufnahmesequenzen erkannt werden. Dazu wird in einem ersten Schritt die Kreuzkorrelation ausprobiert. Wichtig ist, dass das Ergebnis eher zu viele Treffer ausgibt, sprich es sollen eher Rufe ausgegeben werden, die nach genauerer Analyse keine Waldschnepfenrufe sind (false positives). Es sollen jedoch möglichst alle false negatives vermieden werden. False negatives sind Treffer, die richtig sind, aber vom Algorithmus nicht als solche erkannt werden. Die false positives können durch eine Feinanalyse, mit aufwendigeren Berechnungen, aussortiert werden. False negatives sind jedoch irreparabel verloren.

Ziel 2 : Entweder werden Plugins für BatScope geschrieben oder es wird eine schlanke und intuitive Software entwickelt. Welche Richtung eingeschlagen wird, zeigt sich im Laufe der Arbeit, sobald Diskussionen mit der WSL und der Vogelwarte Sempach stattgefunden haben.

Ziel 3 : Die Feinanalyse zeigt möglichst viele false positives.

Ziel 4 : Die Parameter nach Hoodles et al. [3] müssen aus den gefundenen Rufen extrahiert und abgespeichert werden.

2.4 BatScope – eine Zukunftsperspektive

Im Laufe dieser Bachelorarbeit wurde seitens der Vogelwarte Sempach eine Zusammenarbeit mit der WSL diskutiert. Die WSL ist die Eidgenössische Forschungsanstalt für Wald, Schnee und Landschaft. Sie haben bereits eine Software entwickelt, die ähnliche Anforderungen erfüllt. Ihre Software BatScope wird für die Klassifizierung von Fledermäusen eingesetzt. Zu Beginn der Bachelorarbeit stand eine Integration der Algorithmen für die Detektion von Waldschnepfenrufen in BatScope zur Diskussion. Im Laufe dieser Arbeit verlagerte sich der Fokus jedoch von der Integration der zu entwickelnden Algorithmen in BatScope hin zu einer Eigenentwicklung, welche jedoch nicht den gleichen Funktionsumfang aufweist wie BatScope. In Zukunft wird unter Umständen ein AnyScope entstehen, in welches die entwickelten Algorithmen integriert werden können.

3 Ruferkennung

Im Folgenden wird die Erkennung von in einer Audiodatei enthaltenen Balzrufen erläutert. Von der Charakterisierung des Rufs führt der Weg über ausgewählte Methoden der Signalverarbeitung bis hin zu einer Liste von Auftretenszeitpunkten, die durch verschiedene Qualitätsgütern in ihrer Relevanz bewertet werden. Dabei ist wichtig zu betonen, dass die effektive Auswahl der Rufe durch einen erfahrenen Vogelkundler erfolgt. Die hier beschriebenen Verfahren dienen ausschliesslich dazu, Kandidaten vorzuschlagen und zu bewerten. Eine kurze Übersicht soll die angeschnittenen Themenbereiche vorstellen:

In Kapitel 3.1 wird der Balzruf, für den wir uns interessieren, beschrieben. Wir werden die charakteristischen Elemente und Merkmale des Rufs kennenlernen und in Kapitel 3.1.1 erfahren, wie variabel diese Balzrufe eigentlich sind und anhand welchen Merkmalen sie mit den Methoden der Signalverarbeitung erkannt und aufgespürt werden können. Eine Auswahl an Methoden bietet Kapitel 3.1.2. In Kapitel 3.2 werden wir das Grundprinzip unserer Detektion kennenlernen, den Mustervergleich anhand einer Wissensbasis. Wir werden auf die Vorbereitung der Wissensbasis eingehen und erkennen, dass zur Optimierung jetzt bereits bestimmte Berechnungen einmalig durchgeführt werden können. In Kapitel 3.2.1 wird die Erkennung einzelner Flanken thematisiert, in Kapitel 3.2.2 wird gezeigt, wie die erkannten Einzelflanken zu Paaren gruppiert werden können, die einen Pfuiruf beschreiben. Diese beiden Schritte stellen die sogenannte „Vorselektion“ dar. Gefolgt wird sie von den Bewertungsalgorithmen, die eine Qualitätsaussage über die gefundenen Kandidaten abgeben. Wir werden auf zwei Gruppen von Bewertungsalgorithmen eingehen. Zum einen in Kapitel 3.2.3 auf die *Spektrogrammtests*, die bestimmte Informationen aus den Spektrogrammdarstellungen des Rufs nutzen, und zum anderen in Kapitel 3.2.4 auf die sogenannten *Radontests*, die anstelle des Spektrogramms die Radon-Transformierte untersuchen. Die erhaltenen Kandidaten inklusive ihren Bewertungen werden im Frontend schliesslich dem Bediener zur Auswahl angezeigt.

3.1 Rufcharakterisierung

Der Balzruf der Waldschnepfe besteht, wie bereits erwähnt, aus zwei verschiedenartigen Rufkomponenten. Das sogenannte „Quorren“, ein mehrfach wiederholter, tiefer, Gurr-ähnlicher Laut leitet den Balzruf ein. Gefolgt wird er von einem einmaligen auf- und wieder absteigenden Tschirp-Laut, in Fachkreisen „Pfuizen“ genannt, der aufgrund seiner hohen spektralen Bandbreite deutlich robuster nachzuweisen ist als der Quorrlaut. Er durchquert eine Region von unter 4 bis über 8, manchmal bis 13 kHz.

Das Pfuizen ist unter den Vogel-Lauten ein vergleichbar einfach zu detektierender Ruf. Seine im Frequenzspektrogramm unverkennbare bogen-ähnliche Form teilt er nur mit sehr wenigen Lautäusserungen anderer Individuen. So ist bekannt, dass das *Rotkehlchen*, die *Heckenbrunelle* oder die *Singdrossel* einen vergleichbaren Laut in ihrem Repertoire besitzen. Da sie jedoch territorial von der Waldschnepfe unterschieden werden können, lässt sich das Risiko einer Verwechslung in Grenzen halten.

Unter diesen Bedingungen macht es durchaus Sinn, für eine Erkennung des Balzrufs sich mit Methoden der Signalverarbeitung dem Pfuizlaut zu nähern und die Betrachtung des Quorrlauts bis auf Weiteres zurückzustellen.

3.1.1 Variabilität des Pfuizlautes

Der oben beschriebene Pfuizlaut kann, sowohl zwischen Individuen, als auch unter nachfolgenden Rufen desselben Individuums in gewissem Mass variieren. Im Frequenz-Spektrogramm lassen sich diese Unterschiede besonders gut hervorheben. Grob lässt sich ein Pfuiz in drei Zeitabschnitte unterteilen. Der erste, nachfolgend als „aufsteigende Flanke“ bezeichnet, beginnt mit dem Einsetzen eines sehr gleichmässig ansteigenden Sinustons aus dem Hintergrundrauschen und endet beim Eintritt in die Scheitelregion. Der zweite Zeitabschnitt bezeichnet die unter Umständen mehrgliedrige Scheitelpassage bis zum Eintritt in den gleichmässig absteigenden Sinuston, welcher den dritten Abschnitt ausmacht. Jener endet mit dem Eintritt ins Hintergrundrauschen und wird nachfolgend als „absteigende Flanke“ bezeichnet.

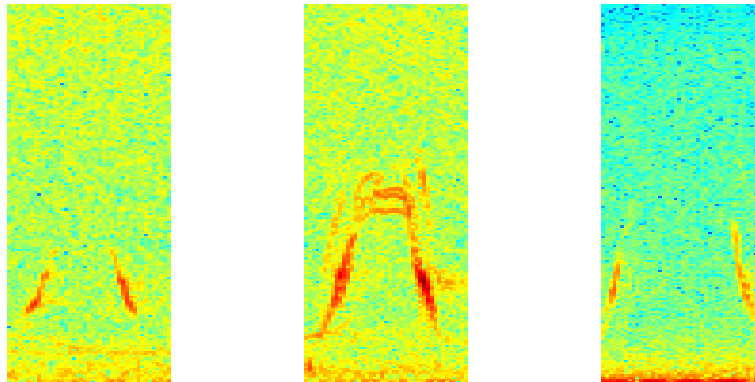


Abbildung 2: Spektra deutlicher Pfuize unterschiedlicher Länge.

Ein erstes variables Merkmal ist die Zeitdauer der auf- resp. der absteigenden Flanke des Rufs (siehe Abbildung 2). Sie kann erfahrungsgemäss, gemessen vom Durchstossen der 5 kHz-Schwelle bis zum Eintritt in den Scheitel, zwischen 0.06 und 0.12 Sekunden

betragen. Dabei ist eine erstaunliche Symmetrie zu beachten, d.h. die Dauer der aufsteigenden Flanke entspricht gerade in etwa jener der absteigenden Flanke. Diese Symmetrie ist so stabil (siehe Abbildung 3), dass sie sogar als Filtereigenschaft verwendet werden kann, um Rufe anderer Arten vom Balzruf der Waldschnepfe zu unterscheiden.

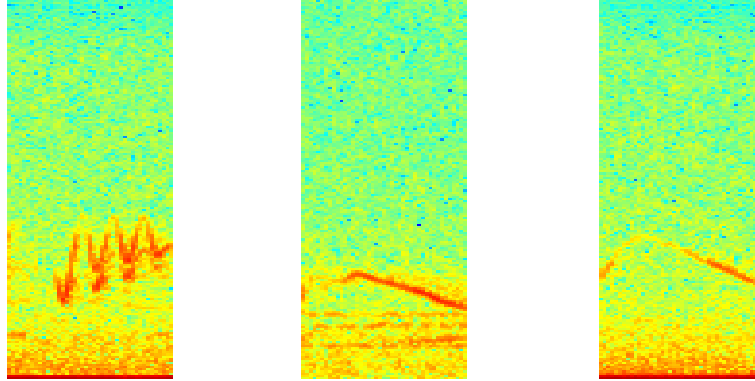


Abbildung 3: Andere, dem Pfuiz ähnliche, aber unsymmetrische Laute.

Ein weiteres variables Merkmal ist die Scheitelfrequenz, als welche die höchste im Ruf vorkommende Frequenz bezeichnet wird (siehe Abbildung 4). Ist bei ihrem Durchtritt kein oder nur sehr wenig Signalenergie vorhanden, so kann es auch nötig sein, sie zu interpolieren. Gemäss den während dieser Arbeit angefallenden Pfuizspektren lässt sich festhalten, dass die Scheitelfrequenz zwischen etwa 7 und 13 kHz. schwanken kann. Höhere Frequenzanteile sind selbst erfahrenen Ornithologen praktisch nicht bekannt.

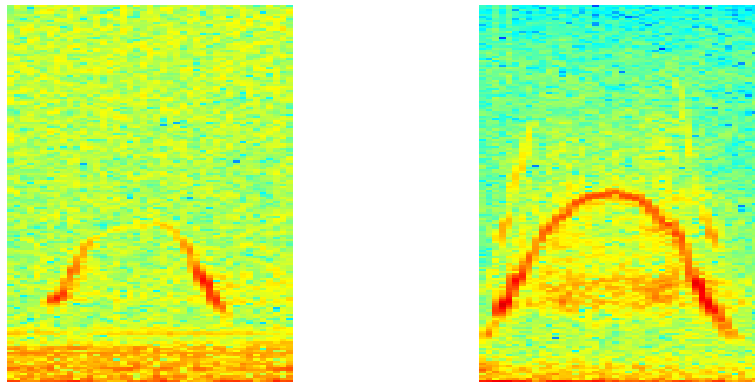


Abbildung 4: Pfuizrufe unterschiedlicher Scheitelfrequenzen.

In der Formausprägung sind vor allem in der Scheitelregion vielfältige Muster zu erkennen (siehe Abbildung 5). Die weitaus häufigste Form ist der einfache, die Flanken schliessende Bogen. Teilweise sind Mehrfachbögen oder kappenartige Abschlüsse zu fin-

den. Speziell ins Auge stechen die geraden, einseitig geneigten, langgezogenen Scheitel, die weit in den tiefen Frequenzen ihren Anfang nehmen können.

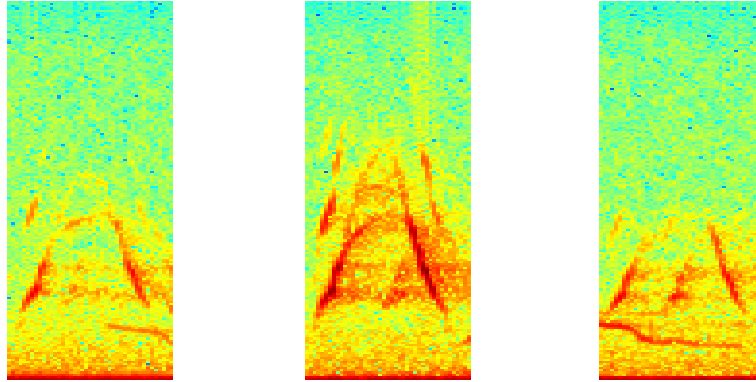


Abbildung 5: Pfuizrufe unterschiedlicher Scheitelausprägungen (desselben Individuums).

Bezüglich der Signalenergie weist ein umfangreicher Vergleich von Pfuiz-Spektrogrammen darauf hin, dass die Flanken im untersten Frequenzbereich vom Individuum mit relativ viel Energie produziert werden, im mittleren Frequenzbereich scheint die Signalenergie jedoch nahezu auf Null abzusinken. Für den Scheitelpartie sind zwei Szenarien feststellbar. In der Mehrzahl der Fälle steigt die Signalenergie wiederum an und produziert im Spektrogramm ein scharfes, klares Abbild der Scheitelpartie. Bei einer kleineren Ansammlung von Pfuizen kehrt die Energie jedoch nicht zurück und der Scheitel wird praktisch stumm durchschritten. Erst in der fallenden Flanke kehrt die Energie zurück (siehe Abbildung 6).

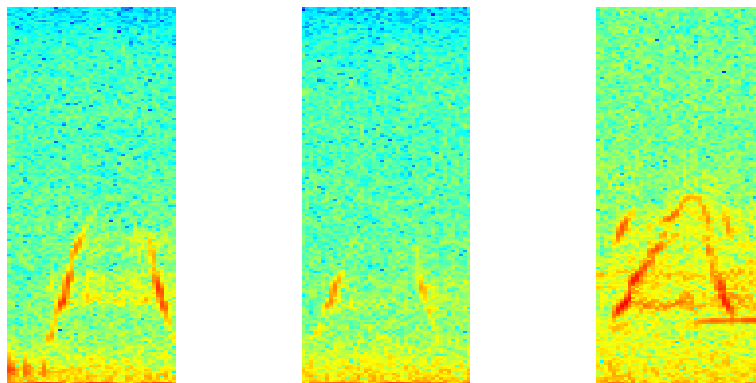


Abbildung 6: Zwei vorherrschende Typen von Energieverteilungen.

Zusammenfassend kann festgehalten werden, dass die Gesamtheit aller Pfuize weder durch eine feste Scheitelfrequenz noch durch vergleichbare fixe Parameter zu lokalisie-

ren ist. Vielmehr ist mit Toleranzbändern zu arbeiten, die die Variabilität der Pfuize gebührend berücksichtigen. Ein solches wird, wie später erläutert, für den zeitlichen Abstand zwischen dem Beginn der steigenden Flanke und dem Beginn der fallenden Flanke am Scheitel, festgelegt. Ebenfalls wird später darauf eingegangen, dass im Zusammenhang mit der Radon-Transformation Toleranzbänder für den Winkel der beiden Flanken verwendet werden.

Eines der sichersten Merkmale letztendlich ist das effektive Vorhandensein der beiden Flanken. Auf diesen Umstand wird sich die Signalverarbeitung in erster Linie stützen.

3.1.2 Merkmale der Signalverarbeitung

Im Zuge der Signalverarbeitung gilt es sich, die Frage zu stellen, wie eine charakteristische Gruppe von Signalen in einer Fülle von unterschiedlichsten Testsignalen aufgefunden werden kann. Die Testsignale können unter Umständen vorgängig zur Kontrolle manuell klassifiziert worden sein. Das Gebiet der *Mustererkennung* hilft in diesem Falle weiter. Es bietet unter anderem Methoden an, die die Übereinstimmung eines Signalausschnitts mit einem anderen numerisch angeben, und zwar so, dass, wenn zwei identische Signale verglichen werden, jeweils ein maximal möglicher Wert resultiert. Sind die Signale verschieden, so resultiert bloss ein „durchschnittlicher“, vergleichsweise geringer Wert. Dabei muss betont werden, dass verschiedene Parameter eines Signals auf Übereinstimmung getestet werden können, so das abgetastete *Zeitsignal*, die *Fourier-Transformierte* davon oder jede vergleichbare, aus den Signalen via Transformation ableitbare Grösse. Im Folgenden wird der Einfachheit halber jedoch immer von „Signal“ gesprochen.

Die bekannteste solcher Methoden nennt sich „Kreuzkorrelation“. Bildlich gesprochen nimmt sie die beiden Signale, berechnet punktweise das Produkt und addiert die Produkte auf, was den numerischen Wert ergibt. Dieses Prozedere wiederholt sie laufend, dabei schiebt sie das Testsignal jeweils um ein Sample weiter am anderen Signal entlang. Sind zwei gleichartige Signale genau übereinander, dann sind alle Produkte maximal und positiv, es gibt keine negativen Werte, über die aufsummiert wird und es resultiert daher der maximal mögliche Wert. Für das mathematisch exakte Vorgehen, das sich die Kreuzkorrelation zunutze macht, sei der Leser auf Kapitel 5.3 im Anhang A verwiesen. Die Kreuzkorrelation in Anwendung wird in Kapitel 3.2 behandelt.

Es stellt sich also die Frage, welche Signale und welche daraus bestimmte Grösse nun sinnvollerweise korreliert werden. Bezüglich Signal ist oben festgestellt worden, dass das Auftreten von relativ homogenen Flanken mit einer unbestimmten Zeitspanne dazwi-

schen jeden Pfuiz zentral ausmacht. Daher wird als Mustersignal mit Sicherheit eine *einzelne steigende* sowie eine *einzelne fallende Flanke* eingesetzt werden. Betreffend die Grösse resp. Funktion aus dem Flankensignal, die zur Korrelation dienen kann, bietet die Signalverarbeitung nach [4] und [5] einige Kandidaten an:

- untransformiert:
 - das Zeitsignal
- via Fourier-Transformation erhalten:
 - Spektrallinien verschiedener Frequenzen
- via Hilbert-Transformation erhalten:
 - die instantane Amplitude
 - die instantane Frequenz
 - die instantane Phase
 - die instantane Energie

Um die Eignung abzuschätzen, wurden verschiedene Tests durchgeführt, sowohl durch einzelne Betrachtung von Auge wie durch automatisierte Prozesse. Signale unter 5 kHz. wurden durch einen Hochpass gefiltert, dies vor allem, um Umgebungsgeräusche, so Kuhglocken, Hundegebell und ähnliches zum Vornherein auszublenden. Dies kommt für die Vergleichssignale, die in einer datenbankähnlichen Struktur permanent vorhanden sind, durchaus in Frage. Das unter Umständen sehr lange Eingangssignal, das überprüft werden soll, kann jedoch ungefiltert bleiben, da die Kreuzkorrelation ein „scharfes“ Ergebnis liefert, sobald eines der Eingangssignale gefiltert ist.

Zwei Graphiken geben nun einen optischen Eindruck einiger Grössen bezogen auf drei ausgewählte Pfuizrufe.

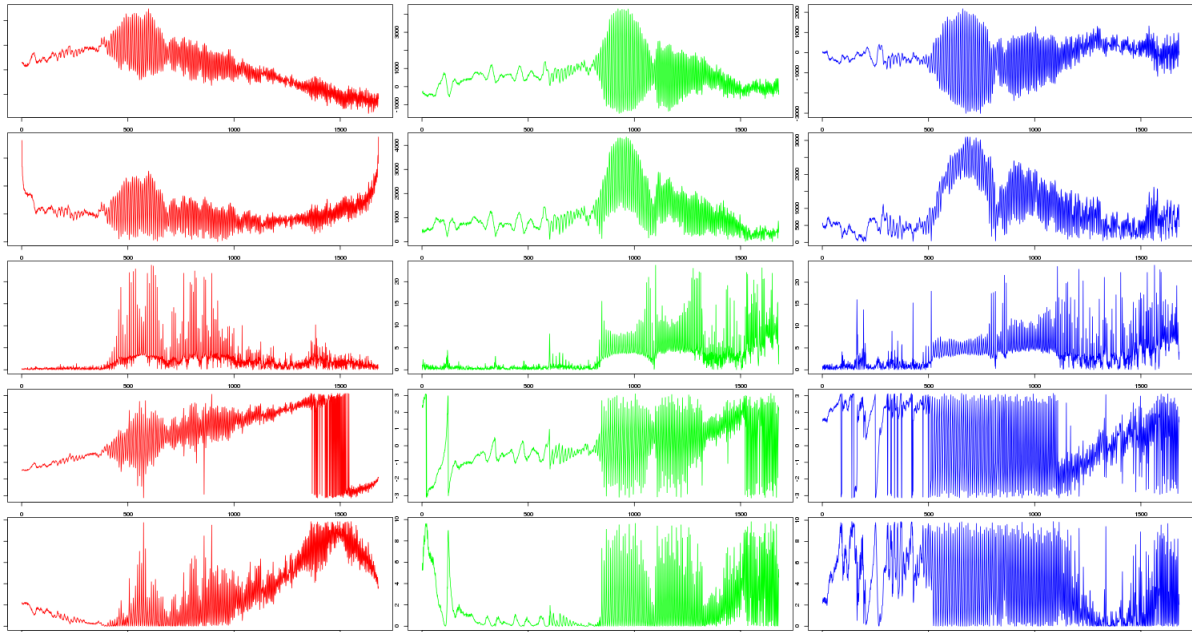


Abbildung 7: *Aufsteigende* Pfuizflanke (von oben nach unten: Zeitsignal / instantane Amplitude / instantane Frequenz / instantane Phase / instantane Energie).

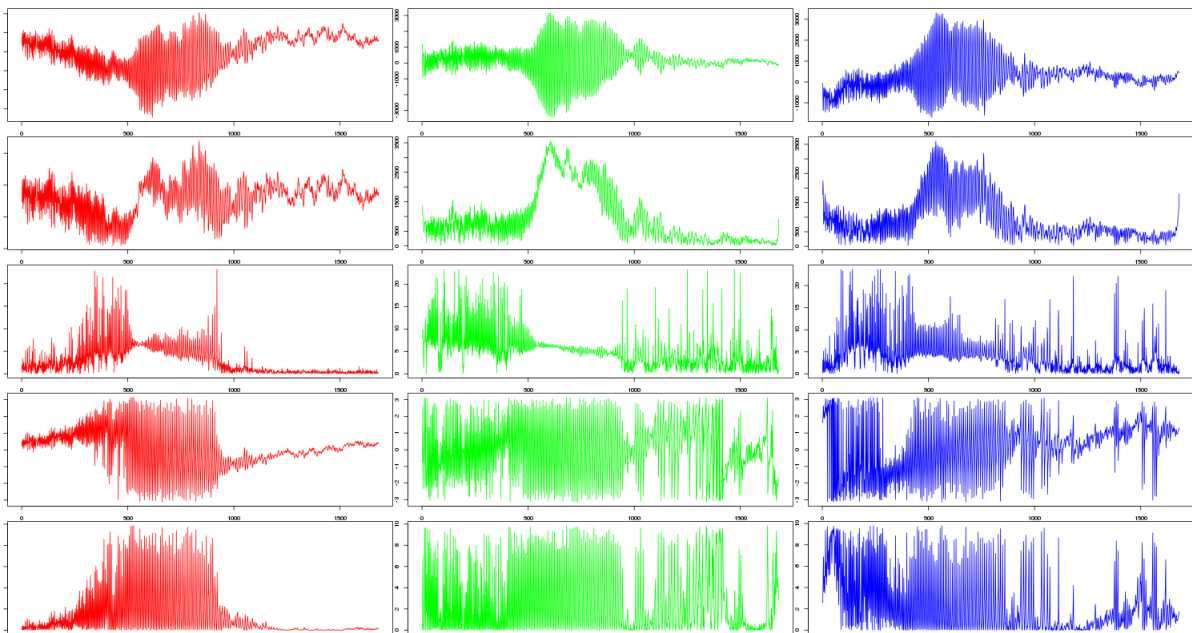


Abbildung 8: *Absteigende* Pfuizflanke (von oben nach unten: Zeitsignal / instantane Amplitude / instantane Frequenz / instantane Phase / instantane Energie).

Wie erkennbar ist, variieren die Grössen zwischen einzelnen Pfuizflanken beträchtlich.

Am ehesten konstant scheinen die obersten zwei Grössen, das *Zeitsignal* sowie die *instantane Amplitude* zu sein. Sie sind in ihrer Eignung tatsächlich auch vergleichbar. In der Tat enthält die Implementation die Korrelation des Zeitsignals. Sein Vorteil ist lediglich das Ersparnis einer weiteren Signaltransformation.

Nach diesem Abschnitt über die Charakteristiken eines einzelnen Rufs wenden wir uns anschliessend nun der Implementation der Ruferkennung in der Software zu, dem Zusammenspiel an verschiedensten Tests, die nötig sind, um über das Vorhandensein zu urteilen.

3.2 Detektions- und Beurteilungsverfahren („Selektion“)

Wie im vorherigen Kapitel angedeutet, arbeitet ein erster Schritt der Ruferkennung mit einer Menge von Pfuizflanken. Diese besteht aus einer einmalig erstellten Sammlung guter Beispielaufnahmen von Pfuizrufen, deren Flanken jeweils separiert und in einer Wissensbasis zusammengetragen werden. Diese Sammlung ist in die zwei Teile „steigende Pfuizflanken“ und „fallende Pfuizflanken“ gegliedert. Dabei ist entscheidend, dass die Beispielaufnahmen in ihrer Form das ganze Toleranzband an möglichen Pfuizflanken *gleichmässig* abdecken. Nur so ist sichergestellt, dass beim durchgeführten Mustervergleich mindestens eine Testflanke als passend erkannt wird. Die Sammlung kann jederzeit um Beispiele ergänzt oder verkürzt werden. Als Vorverarbeitung werden diese Pfuizflanken wie bereits erwähnt bandbegrenzt und mittels eines linearen Filters mit Gauss-Kern (siehe Abbildung 9) dezent geglättet.

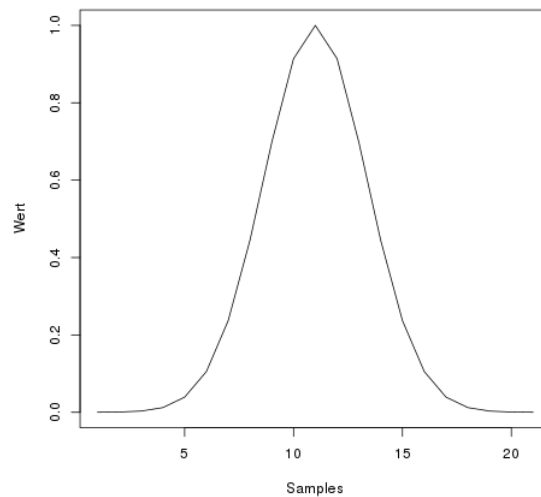


Abbildung 9: Verwendeter Gauss-Filterkern.

Die Darstellung der auf diese Weise erhalten Zeitsignale ist in Abbildung 7, sowie in Abbildung 8 jeweils in der obersten Zeile zu sehen.

Als letzter Schritt der Vorbereitung werden alle diese Pfuizflanken (die steigenden und die fallenden gemeinsam) so auf der Zeitachse ausgerichtet, dass ihre globalen Maxima jeweils beim selben Sample liegen. Das führt dazu, dass bei der Verwendung in der Kreuzkorrelation alle Korrelationsergebnisse der einzelnen Testflanken praktisch am selben Ort zu liegen kommen.

Das eingehende, durchzutestende Signal wird zu allererst in Stücke von 2^{19} Samples zerlegt. Dies ist nötig, da eine solche Audiodatei mehrere Gigabyte an Grösse umfassen kann und nicht davon auszugehen ist, dass sie im Hauptspeicher des Verarbeitungsrechners Platz findet. Das Unterteilen erlaubt zudem ein gutes Parallelisieren des hier beschriebenen Verfahrens auf mehrere Prozessorkerne. Eine Vorfilterung oder Bandbegrenzung wird bei diesem Signal nicht vorgenommen.

Das zu untersuchende Signalstück, im Weiteren schlicht als „Signal“ bezeichnet, wird nun nacheinander mit jedem auf- resp. absteigenden Flankenstück aus der Wissensbasis kreuzkorreliert. Eine graphische Darstellung soll ein solches Korrelationsergebnis verdeutlichen.

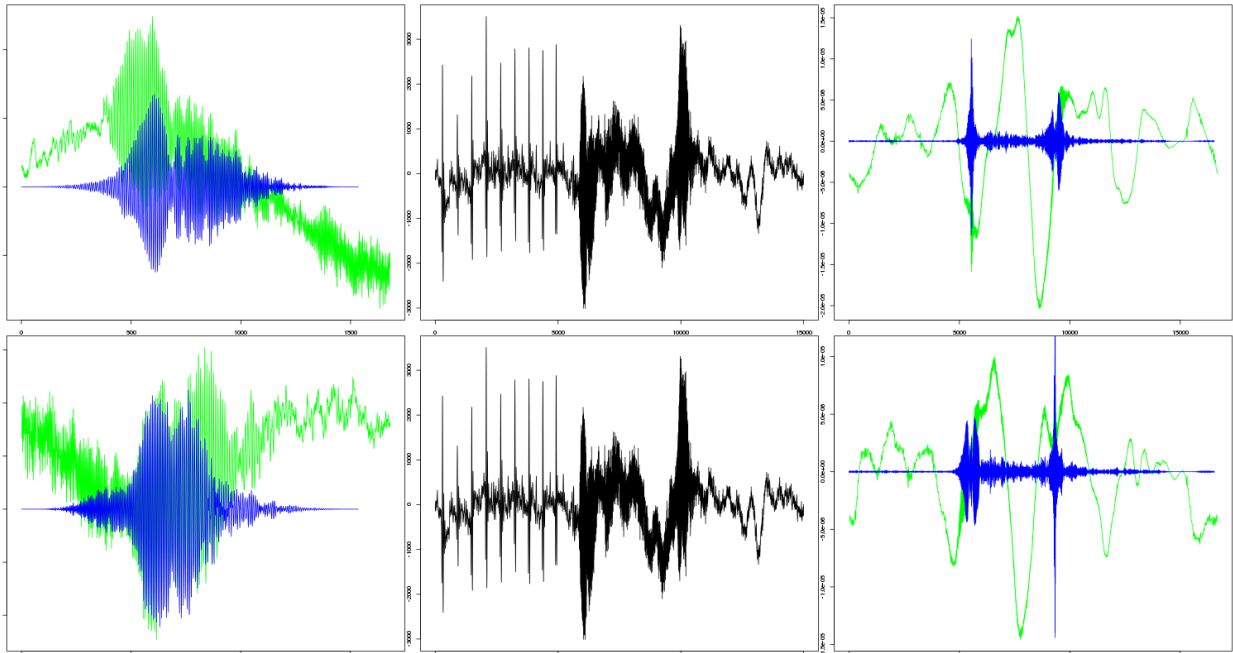


Abbildung 10: Rechts: Korrelationsergebnisse einer steigenden Flanke (oben links) resp. einer fallenden Flanke (unten links) mit einem Pfuizruf (Mitte), in grün die ungefilterte und unverschobene Variante, in blau die nach obiger Beschreibung vorverarbeitete.

Wo nichts anderes vermerkt ist, wird für Beispiele von einer Samplingrate von 48 kHz. bei einer Bittiefe von 16 Bit je Kanal ausgegangen. Bei Stereoaufnahmen wird ausschliesslich der linke Kanal verarbeitet. Die 2^{19} Samples entsprechen so etwa 10,92 Sekunden.

Bemerkenswert dabei ist, dass die steigende Pfuizflanke mit der steigenden Flanke des Signals ein besseres Korrelationsergebnis zeigt als mit der fallenden Flanke, und umgekehrt ebenso. Ein gewisse Treffergüte ist jedoch auch mit der jeweils anderen Flanke zu erzielen.

Die oben erwähnte Faltung ist punktweise sehr aufwändig zu berechnen. Durch die Anwendung des Faltungssatzes kann die zirkuläre Faltung auf die Fourier-Transformation zurückgeführt werden, siehe dazu [6]. Dass sich die Kreuzkorrelation auf die zirkuläre Faltung zurückführen lässt, wird deutlich nach einem Studium der verwendeten Programmiersprache „R“, auf die noch eingegangen wird, zusammen mit deren Manpages, die unter [7] verfügbar sind.

Die Fourier-Transformierte wiederum ist durch die Verwendung der berühmten FFTW-Bibliothek aus C wie aus R (Fortran) heraus sehr rasch zu berechnen. Dazu ist von den zu korrelierenden Signalen jeweils die Fourier-Transformierte zu bilden. Vorgängig sind die beiden Signale noch auf die Länge der Summe beider Signale minus ein Sample von

vorher mit Nullen aufzufüllen, genannt „Zeropadding“. Sobald die Länge des zu testenden Signalstücks bekannt ist (sie wird aus diesem Grund immer konstant gehalten), kann für jede Testflanke dieses Prozedere einmalig durchgeführt werden. Die so vorbereiteten Testflanken-Transformierten werden in einer Datenstruktur vorgehalten, damit sie schnell mit den einkommenden Teststücken verrechnet werden können. Am Schluss wird nämlich eine inverse Fourier-Transformation vorgenommen.

Mit dieser Operation ist die Vorbereitung der Wissensbasis abgeschlossen und der nächste Schritt, das Auffinden und Registrieren einzelner Flanken, kann in Angriff genommen werden.

3.2.1 Einzelflankenerkennung

Bereits in Abbildung 10 wurde das mögliche Resultat dieser Lokalisation aufgezeigt. Dabei kann es zwischen den detektierten Resultaten Abweichungen im Bereich einzelner Samples geben, was es nötig macht, die von den verschiedenen Flanken angezeigten Resultate zu Gruppen zusammenzufassen. Jede Gruppe repräsentiert dabei eine Pfuizflanke mit ihrem Zeitintervall, begrenzt durch den niedrigsten und den höchsten Index des Samples, an dem bisher ein Korrelationsmaximum festgestellt wurde. Dabei wird jeder durch eine nächste Testflanke neu dazukommende Zeitpunkt, sofern er genügend dicht am Intervall liegt, in dieses mit aufgenommen und dessen Grenzen werden angepasst. Genügend dicht heisst standardmässig höchstens 0.04 Sekunden entfernt, möglicherweise wird mit zunehmender Erfahrung ein anderer Wert geeigneter hierfür erscheinen. Die Ergebnis-Gruppen der fallenden und der steigenden Flanken werden getrennt aufgezeichnet (siehe Tabelle 1).

[1] 129049 129134	[1] 128735 129106
0.01053 0.01104 0.00968 0.00828	0.02410 0.02249 0.02024 0.01558
0.00905 0.00927 0.01011 0.00984	0.00000 0.00000 0.00000 0.01923
0.00855 0.01070 0.01058 0.00000	0.00000 0.00847 0.00000 0.01120
	0.00000
[2] 280209 280294	[2] 280182 280238
0.04695 0.04923 0.00000 0.03586	0.02680 0.02582 0.02706 0.01566
0.04247 0.00000 0.00000 0.04513	0.00000 0.00000 0.02233 0.02991
0.00000 0.03217 0.00000 0.04881	0.00000 0.02118 0.00000 0.00000
	0.00000
[3] 283768 284143	[3] 283744 284115
0.02729 0.02921 0.02505 0.02075	0.06245 0.06589 0.06006 0.04851
0.00000 0.00000 0.00000 0.00000	0.00000 0.00000 0.00000 0.05660
0.02379 0.00000 0.02715 0.02414	0.00000 0.02806 0.00000 0.02685
	0.00000
[4] 435027 435122	[4] 434903 435212
0.03354 0.03203 0.04183 0.03134	0.01762 0.01649 0.01844 0.01513
0.02853 0.00000 0.00000 0.03669	0.01687 0.01704 0.00000 0.00000
0.00000 0.02323 0.00000 0.03387	0.00000 0.00000 0.01728 0.00000
	0.00000

Tabelle 1: Vier Ergebnisgruppen, wie sie durch die Kreuzkorrelation erkannt werden, links durch 12 aufsteigende Testflanken, rechts durch 13 absteigende. Die Korrelations-Güten sind von links nach rechts den Testflanken zuzuordnen.

Zwei Dinge sollen an dieser Stelle besonders bemerkt sein. Erstens, dass beileibe nicht alle Testflanken zu jeder Gruppe einen Beitrag leisten, siehe („0.00000“) in Tabelle 1 und zweitens, dass sich die Gruppen 2 und 3 zeitlich sehr nahe beieinander befinden. Dies wird in Kürze noch von Belang sein.

3.2.2 Gruppierung der Einzelflanken

Nach dem Auffinden der Einzelflanken geht es in einem nächsten Schritt darum, die Flanken so zu Gruppen zusammenzufassen, dass sie einen ganzen Pfuizruf beschreiben. Aus der anfänglichen Betrachtung dessen ging hervor, dass die beiden Flanken in einem bestimmten Zeitabstand einander folgen sollen. Dieser hat sich experimentell bei 0.06 bis

0.12 Sekunden als am aussagekräftigsten erwiesen, was vom Ornithologen bestätigt wurde. Für diese Betrachtung wird jeweils das arithmetische Mittel der beiden Randzeiten einer Flankengruppe verwendet.

Es werden nun also alle Paare von Flankengruppen gesucht, so dass die erste Flanke aus der Menge der steigenden Flankengruppen und die zweite aus der Menge der fallenden entstammt und darüberhinaus deren arithmetische Mittel im Intervall $[0.06; 0.12]$ Sekunden liegen. Alle auf diese Weise gefundenen Übereinstimmungen werden provisorisch zu Pfuizkandidaten erklärt und weiter untersucht. Aus dem oben gezeigten Beispiel ergibt dies die folgende Liste 2

-
-
- [1] 125299.0 128920.5 (Gruppe [1] als absteigende Flanke)
 - [2] 280251.5 283929.5 (aus Gruppen [2] und [3] gebildet)
 - [3] 435074.5 438887.0 (Gruppe [4] als aufsteigende Flanke)
-
-

Tabelle 2: Drei Pfuiz-Gruppen, in denen obige vier Flanken-Gruppen als Teile vorkommen, die Pfuiz-Gruppe 2 ist vollständig aus den Flanken-Gruppen 2 und 3 gebildet worden.

Die auf diese Art erhaltenen Pfuiz-Kandidaten sind aber noch nicht „rein“ genug. Z.B. könnten während eines Regenwetters aufprasselnde Tropfen im richtigen Zeitabstand (was so gut wie sicher vorkommt) die bisherigen Tests immer noch bestehen. Deshalb folgt als nächstes ein Test, der sich das Spektrogramm des ausgewählten Zeitfensters näher betrachtet. Zum Zeitbereich ist zu sagen, dass obige Zeiten den durchschnittlichen Korrelationszeitpunkt, also den Zeitpunkt, an dem die Korrelation ihr Maximum annimmt, beschreiben, und dieser sich immer am Anfang des getesteten Signals befindet. Soll nun ein Spektrogramm des gesamten Pfuizrufs gebildet werden, so muss das Zeitfenster nochmals um seine Länge erweitert werden. Die Längen werden danach noch nach „aussen“ hin gerundet. Obige effektive Zeitfenster ergeben sich also zu 3:

-
-
- [1] 125299.0 128920.5, effektiv: [125299; 132542]
 - [2] 280251.5 283929.5, effektiv: [280251; 287608]
 - [3] 435074.5 438887.0, effektiv: [435074; 442700]
-
-

Tabelle 3: Effektive, auf das Doppelte verlängerte Zeitfenster

3.2.3 Spektrogrammtests

Von diesen erhaltenen Zeitintervallen kann nun ein Spektrogramm erstellt werden. Der für die Betrachtung von Pfuizrufen interessante Spektralbereich erstreckt sich von 5 bis 13 kHz. Das Spektrogramm muss in einer ausreichenden Auflösung hergestellt werden, damit seine Daten auch für die auf der Radon-Transformation basierenden Tests verwendet werden können und nicht erneut ein Spektrogramm berechnet werden muss. Bei 48 kHz. Samplingrate wird im Moment eine Fensterlänge von wahlweise 128 resp. 256 Samples verwendet, auf eine Überlappung der Fenster wird verzichtet.

Wie in Kapitel 3.1.1 ausgeführt, sind Pfuiz-Spektrogramme weitestgehend monoton. Lediglich im Scheitelpunktbereich kann es Abweichungen davon geben. Aus diesem Grund scheint es vernünftig, die Struktur im Spektrogramm in erster Linie auf ihr Monotonieverhalten zu testen. Dazu werden in den vorberechneten Frequenzanteilskurven, die die Amplitude einer bestimmten Frequenz über die Zeit darstellen, die zwei im gesamten Zeitintervall stärksten Maxima gesucht und deren Sampleposition relativ zum Beginn des Zeitintervalls für jede einzelne Frequenz festgestellt. Bei einer monotonen Struktur mit zwei Flanken werden die Maxima zu höheren Frequenzen hin gegen die Mitte des Zeitintervalls „wandern“. D.h. das erste der beiden Maxima sollte mit jeder höheren Frequenz etwas später auftreten, das zweite jedoch etwas früher. Am Scheitel können die Frequenzmaxima möglicherweise zusammenfallen. Es wird nun gezählt, wie oft diese Bedingung effektiv erfüllt ist, und wie oft sie maximal erfüllt sein könnte, wenn ein perfekter Pfuiz mit Scheitelfrequenz 13 kHz. auftreten würde.

Ein in der Software einstellbarer Parameter erlaubt es nun, verschiedene Beurteilungsverfahren für die Monotonie-Qualität zu wählen. Ausgehend von der maximalen Anzahl möglicher erfüllter Schritte wird eine Punktzahl festgelegt. Je nach gewähltem Verfahren wird eine Verletzung der Bedingung durch Abzug eines Punktes oder durch die Nicht-Erteilung des entsprechenden Punktes quittiert. Auch kann angegeben werden, ob die Monotonie schwach oder stark gewertet werden soll. So sind auch Mischverfahren denkbar, die bei starker Monotonie einen Punkt vergeben, bei schwacher Monotonie keinen und bei Verletzung der Monotonie einen entziehen. Ein vorbildlicher Pfuiz mit den zwei erkannten Maxima pro Frequenzlinie ist in Abbildung 11 ersichtlich.

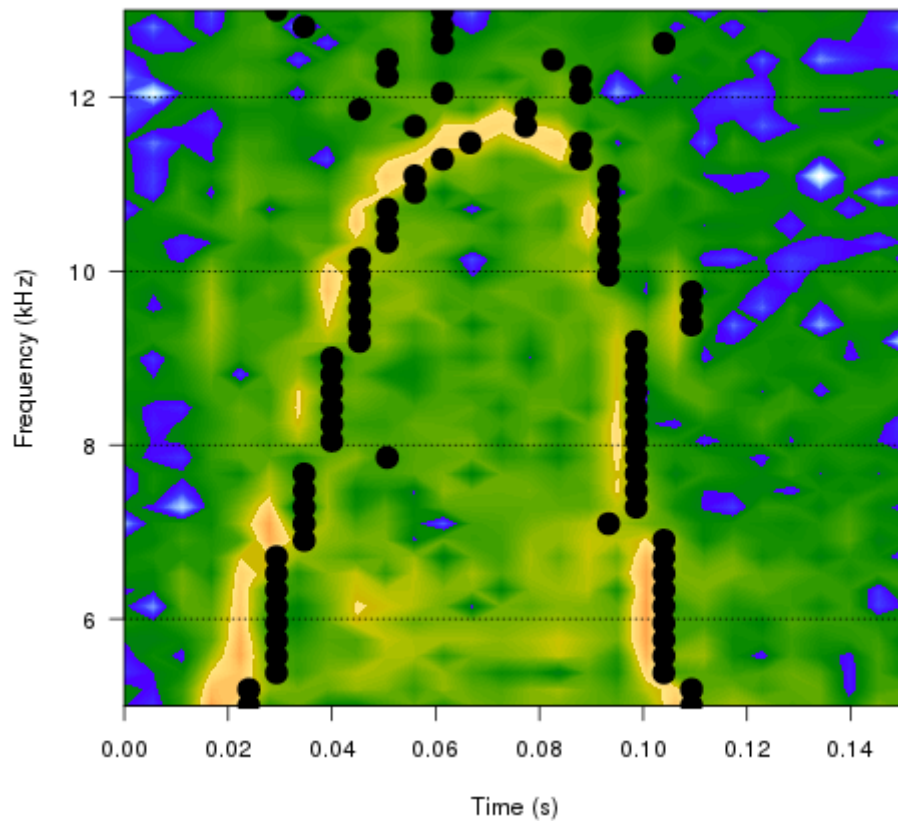


Abbildung 11: Die Monotoniebedingung ist hier relativ gut erfüllt.

Es ist hier festzustellen, dass oberhalb des Scheitels die Maxima natürlich nicht mehr an den Pfuizruf gebunden sind und daher mehr oder weniger zufällig zu liegen kommen. So wird ein Pfuizruf mit sehr niedriger Scheitelfrequenz bei diesem Test erwartungsgemäss schlechter abschneiden als ein Ruf mit hoher Scheitelfrequenz.

Daher ist komplementär zum Monotonietest noch ein weiterer Test eingeführt worden, der die Tatsache berücksichtigt, dass die zeitlichen arithmetischen Mittel zwischen den beiden Maxima je einer Frequenzlinie im Falle eines Pfuizes eine Häufung in einer bestimmten Zeitregion zeigen. Die nachfolgende Abbildung 12 soll dies verdeutlichen.

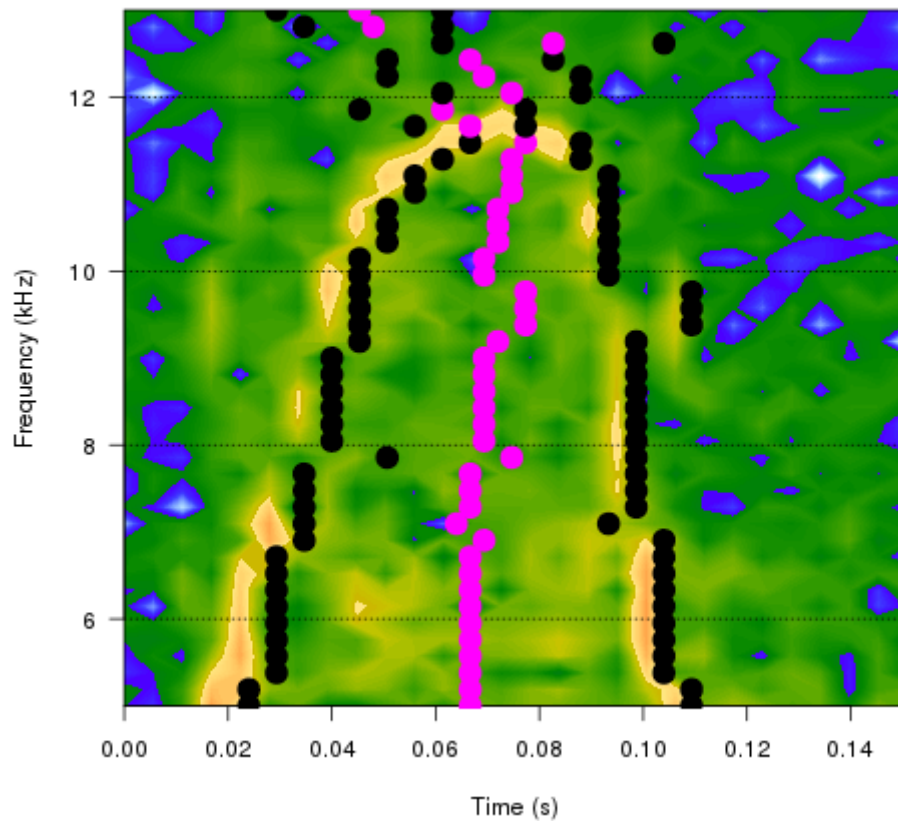


Abbildung 12: Derselbe Pfuiiz mit eingezeichneten Mittelpunkten zwischen den beiden Maxima je Frequenz.

Da die beiden Maxima auf dem gesamten Zeitintervall gesucht werden, kann auch der Mittelpunkt im gesamten Zeitintervall zu liegen kommen, hat also potenziell den selben zeitlichen Wertebereich. Für ein Rauschsignal wird sich seine Varianz jedoch halbieren gegenüber der Varianz der einzelnen Maxima. Liegt nun ein Pfuiizruf vor, so häufen sich die Maxima massiv an einer Stelle und die Varianz sinkt. Selbst bei niedrigerer Scheitelfrequenz wird dies spürbar hervortreten. Um einen besseren Eindruck über die Verteilung der arithmetischen Mittelwerte zu erhalten, ist in der nachfolgenden Abbildung 13 ein entsprechendes Histogramm eingefügt worden.

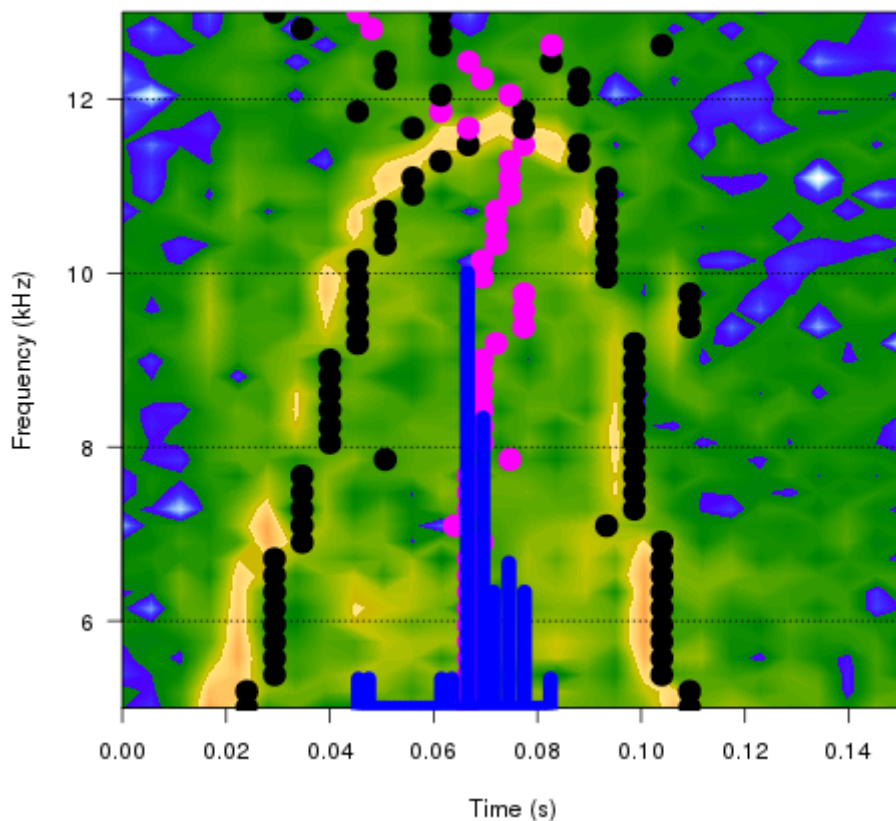


Abbildung 13: Eine dritte Variante mit Histogramm der in violett dargestellten Mittelpunkte.

Der mittels Histogramm erhaltene sogenannte „Modus“, also die Anzahl Treffer beim Zeitabschnitt mit den meisten Mittelpunkstreffern, muss im *Modustest* nun bezüglich der Anzahl aller Mittelwerte einen gewissen relativen Anteil aufweisen, damit der Test als bestanden gilt. Alternativ liesse sich der Test durch Betrachtung der Prominenz, also der Stärke des „Herausstechens“ des Wertes aus seiner Umgebung, noch weiter verfeinern. Aus diesem relativen Anteil wird, wie übrigens auch bei der Monotonie, eine Güte im Intervall $[0; 1]$ (oder je nach Vefahren in $[-1; 1]$) bestimmt und mit jener zum Resultat der Spektrogrammgesamtttests zusammengefasst. Der Spektrogrammttest nimmt selbst auch eine Bewertung anhand der beiden Güten vor. Der Zeitabschnitt selbst, der die meisten Treffer aufweist, wird formal zum „Zentrum“ des Pfuizes erklärt und spielt in den *Radontests* eine wichtige Rolle. Eine Gesamtbewertung der Spektrogrammttests enthält die nachfolgende Tabelle 4 und kann von der Softwarelogik zur finalen Klassifizierung

unverbindlich verwendet werden.

```
[1] quality.monotony 0.7560976
[2] result.monotony TRUE
[3] quality.histogram 0.3571429
[4] result.histogram TRUE
[5] center.time 128474
```

Tabelle 4: Resultat der Spektrogrammtests.

Dem letzten Test aus der Gruppe der Bewertungsalgorithmen, dem Radon-Test, wird das Kapitel 3.2.4 gewidmet.

3.2.4 Radon-Transformation

Die Radon-Transformation[8] soll zum einen bestätigen, dass ein durch Korrelation gefundener Ruf tatsächlich einer ist und zum anderen sollen damit Parameter nach Hoodles et al. ausgemessen werden. Die Radon-Transformation kommt aus dem Bereich der Computertomographie und findet auch im Bereich der Bildverarbeitung Anwendung.

Funktionsweise Die Definition der Radon-Transformation ist in 5.7 zu finden. Grundsätzlich werden die Pixelwerte eines Bildes aufsummiert. Jedoch nicht über das ganze Bild gleichzeitig, sondern nur entlang von Geraden, die über das Bild gelegt werden. Es wird für jeden Winkel von 0 bis 180 Grad und vom Mittelpunkt bis zur Bildecke (grösster Abstand zum Bildmittelpunkt) eine Gerade über das Bild gelegt und die Pixelwerte werden aufsummiert. Die Parameter für die Gerade sind jeweils auch die Koordinaten, in welcher das Resultat der Radon-Transformierten zu liegen kommt. Dieser Koordinate wird, abhängig vom Ergebnis der Radon-Transformation, ein Pixelwert zugeordnet. Grosse Summen ergeben hellere Pixelwerte und kleine Summen dunklere Werte. In der Radon-Transformierten kann überprüft werden, ob Flanken eines Pfuizes an einem bestimmten Ort mit einem bestimmten Winkel vorhanden sind.

Analysebereich Für die Radon-Transformation wird nur ein Ausschnitt aus dem Spektrogramm des Originalbildes verwendet (siehe Abbildung 14). Der Bereich wurde so gewählt, dass möglichst nur die Pfuiz-Flanken transformiert werden. Dies ist sehr wichtig, da ansonsten das Ergebnis der Radon-Transformation durch Signale in anderen

Frequenzbereichen verfälscht wird. Der Pfuiz sollte auf dem Spektrogramm bereits zentriert sein. Dadurch darf der Parameter „Abstand zum Mittelpunkt“ als relativ konstant angenommen werden. Ansonsten würde sich dieser Bereich, je nachdem wie der Pfuiz auf dem Spektrogramm liegt, in der Radon-Transformierten nach unten oder nach oben verschieben. Zudem ist der Ausschnitt des Spektrogrammes, der transformiert wird, immer gleich gross. Sonst wäre die Radon-Transformierte jeweils unterschiedlich gross. Die Höhe der Radon-Transformation ist abhängig von der Grösse der Bilddiagonalen des Originalbilds.

Der Analysebereich ist kein einzelner Punkt, sondern ein für beide Parameter definierter Bereich, da nicht alle Pfuize gleiche Abstände und Flankensteigungen aufweisen. In der Radon-Transformierten ergibt sich daraus pro Pfuiz-Flanke jeweils ein Rechteck (siehe Abbildung 15). Bei der Untersuchung von mehreren Radon-Transformierten hat sich ergeben, dass linke Pfuiz-Flanken jeweils zwischen 165 und 178 Grad hohe Intensitäten haben und rechte Pfuiz-Flanken jeweils zwischen 5 und 18 Grad. Beim zweiten Parameter „Abstand zum Bildmittelpunkt“ hat sich ergeben, dass Werte zwischen 40 und 50 gut funktionieren.

Anwendung - Feinanalyse für Aussortierung von „False Positives“ Wenn keine Flanken vorhanden sind, sind alle Werte in diesem Bereich etwa gleich. Sind in diesem Bereich Flanken vorhanden, dann sollte sich eine Linie durch diesen Bereich ziehen. Während dieser Bachelorarbeit wurden mehrere Grössen ausprobiert, welche angeben, ob eine Linie durch diesen Bereich führt. Die Standardabweichung hat dabei mehrheitlich gute Resultate geliefert. Wenn auf dem Bildausschnitt nur Rauschen sichtbar ist, ist die Standardabweichung kleiner als wenn eine klar sichtbare Linie durch den Bildausschnitt geht.

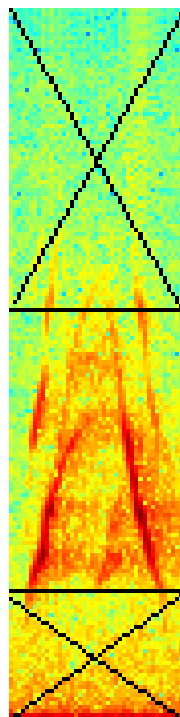


Abbildung 14: Spektrogramm eines Pfuiiz mit Angabe des gefilterten Bereiches.

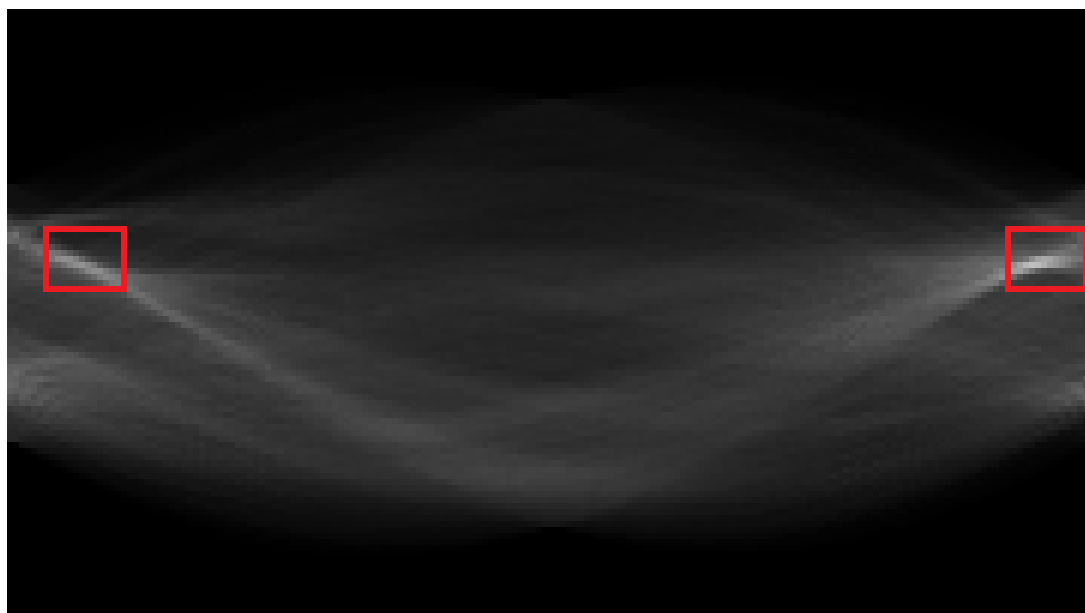


Abbildung 15: Radon-Transformierte des gefilterten Spektrogrammes aus der Abbildung oben. Die rot markierten Bereiche werden ausgewertet.

Anwendung - Berechnung Hoodles Parameter Die Hoodles Parameter wurden anhand des Maximums innerhalb des Rechteckes für die Flanken berechnet. Die Position

des Maximums innerhalb der beiden Rechtecke auf der Radon-Transformierten gibt jeweils die Gerade an, die auf den beiden Pfuiz-Flanken liegen. Diese Geraden werden an den Stellen 3.5 kHz und 7.5 kHz ausgewertet. Bei Hoodles et al. [3] sind die Parameter D1 und D2 jeweils der Abstand zwischen linker Flanke und rechter Flanke bei 3.5 kHz und bei 7.5 kHz. Daraus können die Parameter D3 und D4 berechnet werden, welche jeweils die Steigungen der Flanken darstellen.

Vor- und Nachteile Die Radon-Transformation hat gegenüber der sonst üblichen Methode der Hough-Transformation Vorteile. Sie verarbeitet nur binäre Bilder. Die Radon-Transformation hingegen hat diese Einschränkung nicht. In der Hough-Transformation wird meist zuerst eine Kantendetektion durchgeführt, damit die Kanten beim Binarisieren übrig bleiben. Beim Pfuiz ist diese Kantendetektion nicht hilfreich, da beim Bogen des Pfuiz zwei Kanten entstehen. Der äussere und der innere Bogen. Für das Binarisieren wird meist ein Schwellwert definiert. Bei Aufnahmen, welche von unterschiedlichen Aufnahmegeräten und unterschiedliche Aufnahmeeinstellungen aufgenommen wurden, ist ein fixer Schwellwert etwas problematisch, da dieser meist auf Basis der verfügbaren Aufnahmedaten festgelegt wurde. Die Radon-Transformation hingegen hat dieses Problem nicht, da sie für die Transformation kein binarisiertes Bild benötigt. Es werden alle Signalanteile für die Transformation verwendet. Der Nachteil der Radon-Transformation ist im Gegenzug, dass ihr Ergebnis nicht so klar ausfällt wie das der Hough-Transformation.

3.2.5 Verifizierung der verwendeten Features

Die Abbildung 16 stellt die verwendeten Features zur Trennung zwischen den beiden Klassen Pfuiz und kein Pfuiz dar. Es zeigt sich, dass die Kombination der verwendeten Features die beiden Klassen relativ gut trennt. Als einzelnes Feature wäre die Monotonie am besten. Durch die gemeinsame Betrachtung mehrerer Features ergibt sich möglicherweise eine bessere Trennung, da die verschiedenen Features die beiden Klassen unterschiedlich gut trennen.

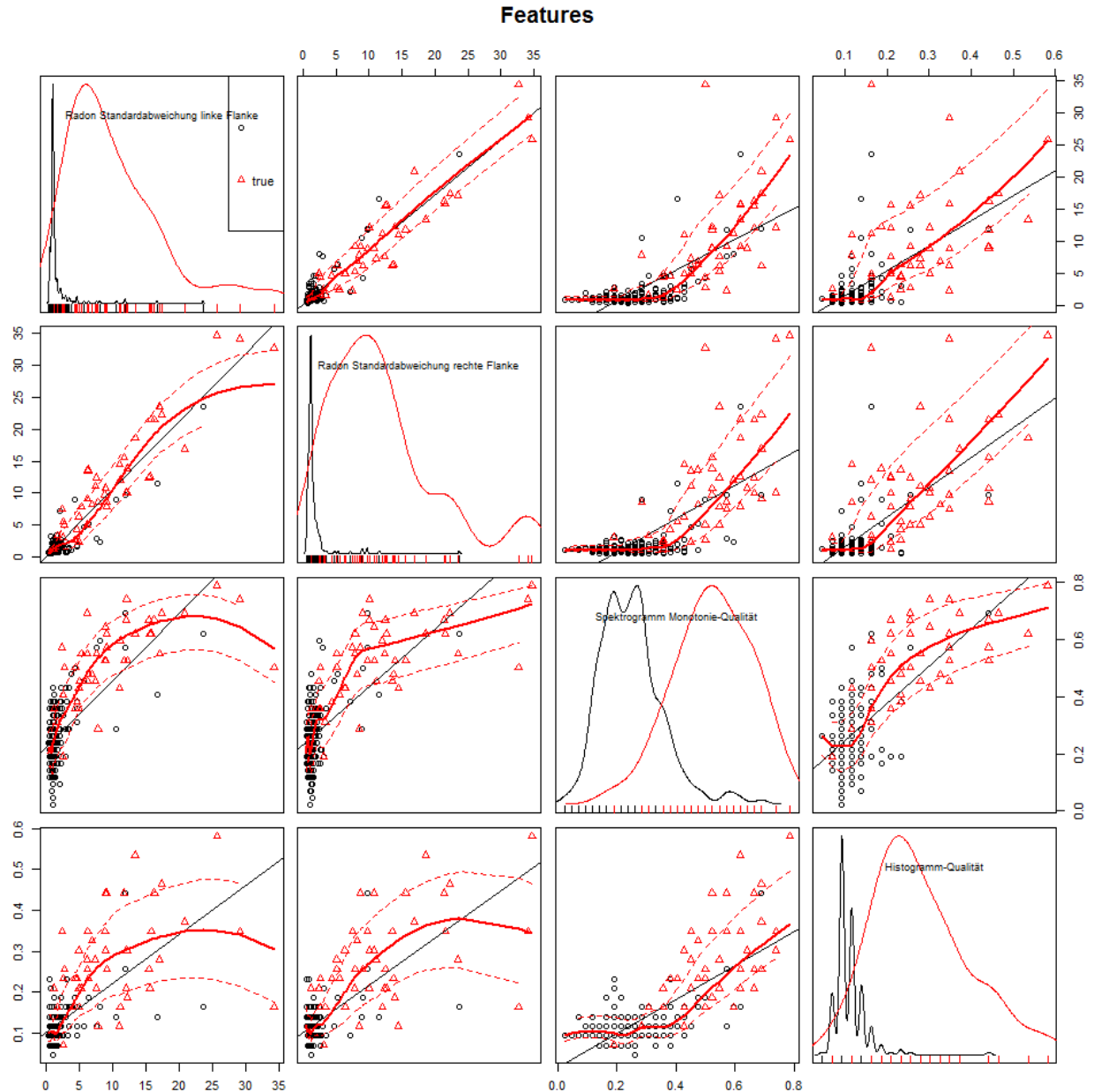


Abbildung 16: Regressionsplots der verwendeten Features. Rote Dreiecke sind Pfuize und schwarze Kreise sind keine Pfuize.

3.3 Demonstration der Fähigkeiten – Benchmark

Nachdem nun alle algorithmischen Verfahren vorgestellt worden sind, dient dieser Abschnitt dazu, anhand einer realen Beispielaufnahme die erlangten Fähigkeiten unter Beweis zu stellen. Dazu werden alle oben besprochenen Tests und Klassifikationen durch-

geführt und deren Ergebnis verglichen mit einem manuell erstellten Ergebnis eines erfahrenen Ornithologen.

Als Beispielaufnahme soll ein 16,1 Sekunden langer Ausschnitt aus dem Gebiet Amden dienen. Er zeichnet sich dadurch aus, dass sowohl sehr starke, vorbildliche, wie auch fast überhörbare Balzrufe enthalten sind. Daneben enthält die Aufnahme Störgeräusche anderer Vogelrassen sowie Strassengeräusche. Der Hintergrund-Rauschpegel ist ebenfalls nicht zu vernachlässigen.

Ein Plot des Zeitsignals sowie ein Spektrogramm (siehe Abbildung 17) sollen die gegebene Situation darstellen. Die gefundenen, in Wahrheit zutreffenden Rufe sind in grün markiert (True Positives), in rot die nicht entdeckten (False Negatives) und in gelb die Kandidaten, die sich als Geräusch anderer Art (False Positives) herausstellen.

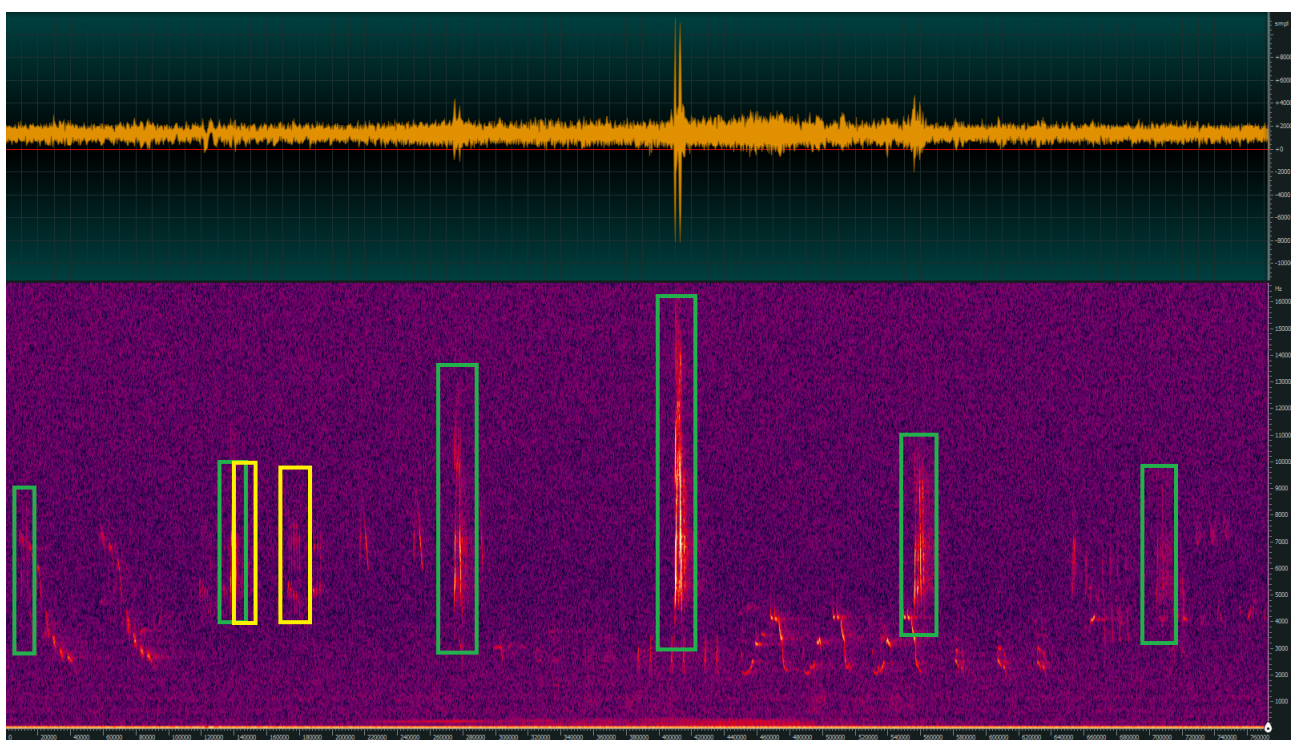


Abbildung 17: Zeitsignal und Spektrogramm der Benchmark-Aufnahme.

Zwei bemerkenswerte Ausschnitte, der schwächste und einer der stärksten Pfuizlaute der Aufnahme, seien dabei exemplarisch hervorgehoben (siehe Abbildung 18).

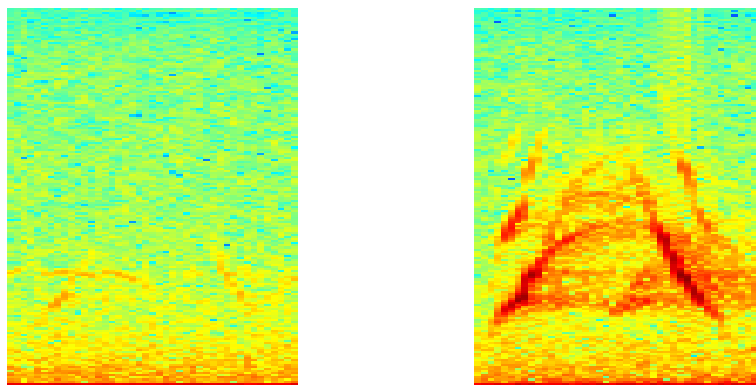


Abbildung 18: Zwei Pfuizrufe unterschiedlicher Qualität.

Gemäss Kapitel 3.2 stellt der erste Schritt die Vorbereitung der Wissensbasis dar. Sie läuft im Verborgenen ab und erzeugt keine für die Bewertung relevanten Ausgaben. Im Folgenden werden aus Platzgründen speziell diese beiden hervorgehobenen Pfuizrufe weiter untersucht. Die Gesamtheit an Information der anderen Pfuizrufe ist unter Umständen umfangreicher als hier präsentiert.

Die Einzelflankenerkennung gemäss Kapitel 3.2.1 liefert 33 Kandidaten für aufsteigende Flanken und 34 für absteigende. Wie bereits erkennbar ist, werden sich die hervorgehobenen Rufe aus den aufsteigenden Flanken mit den Nummern 1 und 21, resp. aus den absteigenden Flanken mit den Nummern 2 und 23 zusammensetzen. Die folgende Tabelle 5 soll die Resultate der Einzelflankenerkennung verdeutlichen.

erkannte aufsteigende Flanken:

- erste und letzte korrelierte Position
angegeben im Index des Samples
- Qualitätszahlen der Testsamples liegen
im Intervall [0;1]

```
[[1]]$max.pos.interval
[1] 8559 8634
[[1]]$matching.qualities
[1] 0.001 0.001 0.001 0.001 0.000 0.000
[7] 0.001 0.001 0.000 0.000 0.000 0.000

[[2]] 20390 20475
[[3]] 67248 67267
[[4]] 69411 69833
[[5]] 117531 117541
[[6]] 120912 122481
[[7]] 132299 132883
[[8]] 136610 136710
[[9]] 139779 140162
[[10]] 143381 143438
[[11]] 171688 172216
[[12]] 175138 175233
[[13]] 185469 185544
[[14]] 187489 191279
[[15]] 215558
[[16]] 220195 220205
[[17]] 253312 253456
[[18]] 273637 273737
[[19]] 276844 277399
[[20]] 279756 279765

[[21]]$max.pos.interval
[1] 408655 408827
[[21]]$matching.qualities
[1] 0.031 0.026 0.028 0.032 0.033 0.000
[7] 0.000 0.025 0.000 0.019 0.000 0.033

[[22]] 411603 412290
[[23]] 414637 414756
[[24]] 555064 555149
[[25]] 558545 558960
[[26]] 561003 562861
[[27]] 651816 652474
[[28]] 659778 659872
[[29]] 662641 662698
[[30]] 703313 703407
[[31]] 707027 707286
[[32]] 713244 713263
[[33]] 718726 719622
```

erkannte absteigende Flanken:

```
[[1]] 8550 8597

[[2]]$max.pos.interval
[1] 12131 12141
[[2]]$matching.qualities
[1] 0.000 0.000 0.000 0.000 0.000 0.000
[7] 0.000 0.000 0.000 0.000 0.000 0.000
[13] 0.000

[[3]] 18550 20418
[[4]] 68205 69729
[[5]] 117938 118135
[[6]] 120560 122407
[[7]] 132398 132827
[[8]] 136368 136863
[[9]] 139747 140172
[[10]] 143382 144990
[[11]] 171753 176708
[[12]] 179010 179047
[[13]] 184908
[[14]] 187109 187546
[[15]] 191204 191260
[[16]] 220359 220576
[[17]] 253550 253847
[[18]] 273580 273906
[[19]] 277228 277370
[[20]] 280300 280904
[[21]] 290752 290805
[[22]] 408674 408910

[[23]]$max.pos.interval
[1] 412116 412290
[[23]]$matching.qualities
[1] 0.024 0.030 0.030 0.037 0.000 0.000
[7] 0.042 0.029 0.000 0.016 0.000 0.009
[13] 0.000

[[24]] 414645 414766
[[25]] 554950 555102
[[26]] 558552 558951
[[27]] 560957 563617
[[28]] 651789 652696
[[29]] 659626 659806
[[30]] 662623 662660
[[31]] 703227 703498
[[32]] 707064 707381
[[33]] 713139 713141
[[34]] 718718 719613
```

Tabelle 5: Im Benchmark erkannte Einzelflanken.

Hier wurden die Parameter der Algorithmen derart tolerant gewählt, dass kein Pfuizruf „untergeht“. Ein späteres Ausschliessen eines Störgeräusches ist verglichen mit dem manuellen Auffinden eines vergessenen Rufs recht einfach.

Die Voraussetzungen für ein Gruppieren beider Flanken zu einem einzigen Pfuizruf sind also bereits gegeben. Ein entsprechender Gruppierungstest nach Kapitel 3.2.2 soll in dieser Frage Klarheit schaffen. In der folgenden Tabelle 6 sind die gruppierten Flankenpaare zusammengestellt.

erkannte Gruppen von Flanken => mögliche Pfuizrufkandidaten - linke und rechte Flankenposition in Samples - alternativ zusätzlich in Sekunden	
[[1]]\$samples	[[8]]\$samples
[1] 8596.5 12136.0	[1] 215558.0 220467.5
[[1]]\$seconds	[[8]]\$seconds
[1] 0.1790937 0.2528333	[1] 4.490792 4.593073
[[2]]\$samples	[[9]]\$samples
[1] 117536.0 121483.5	[1] 273687 277299
[[2]]\$seconds	[[9]]\$seconds
[1] 2.448667 2.530906	[1] 5.701812 5.777063
[[3]]\$samples	[[10]]\$samples
[1] 132591.0 136615.5	[1] 277121.5 280602.0
[[3]]\$seconds	[[10]]\$seconds
[1] 2.762313 2.846156	[1] 5.773365 5.845875
[[4]]\$samples	[[11]]\$samples
[1] 136660.0 139959.5	[1] 408741 412203
[[4]]\$seconds	[[11]]\$seconds
[1] 2.847083 2.915823	[1] 8.515438 8.587563
[[5]]\$samples	[[12]]\$samples
[1] 139970.5 144186.0	[1] 555106.5 558751.5
[[5]]\$seconds	[[12]]\$seconds
[1] 2.916052 3.003875	[1] 11.56472 11.64066
[[6]]\$samples	[[13]]\$samples
[1] 175185.5 179028.5	[1] 558752.5 562287.0
[[6]]\$seconds	[[13]]\$seconds
[1] 3.649698 3.729760	[1] 11.64068 11.71431
[[7]]\$samples	[[14]]\$samples
[1] 185506.5 191232.0	[1] 703360.0 707222.5
[[7]]\$seconds	[[14]]\$seconds
[1] 3.864719 3.984000	[1] 14.65333 14.73380

Tabelle 6: Resultat des Gruppierungstests.

Aus den anfänglichen 77 Flanken konnten 14 Gruppen gefunden werden, die passende Zeitparameter aufweisen. Die Rohdaten aus den erkannten Gruppen werden nacheinander den einzelnen Beurteilungsverfahren zugeführt, wie in Kapitel 3.2.3 erläutert.

Im Rahmen der Spektrogrammtests erscheint es am sinnvollsten, lediglich die Spektrogramme der zwei ausgewählten Pfuizrufe mit erkannten Frequenzmaxima und Histogrammen

sowie als Kontrast dazu ein Spektrogramm eines Störgeräusches darzustellen (siehe Abbildungen 19, 20 und 21).

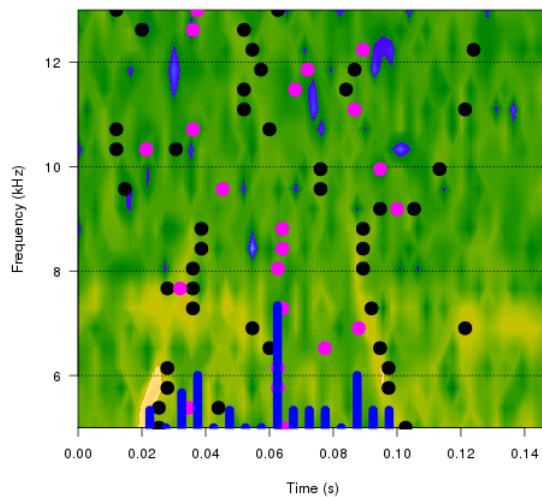


Abbildung 19: Spektrogramm des schwächsten Pfuirufs mit Zeitmarke 11696.

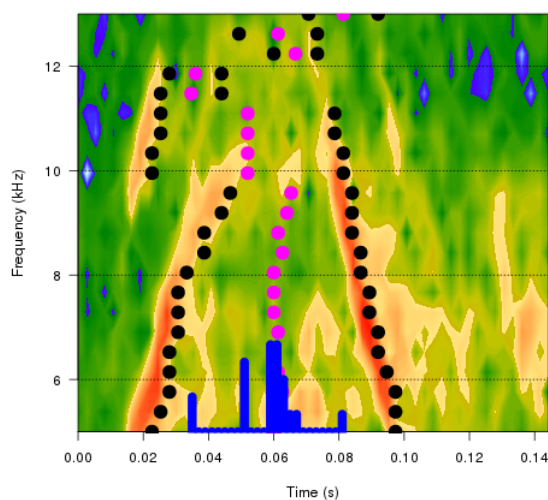


Abbildung 20: Spektrogramm eines starken Pfuirufs mit Zeitmarke 411691.

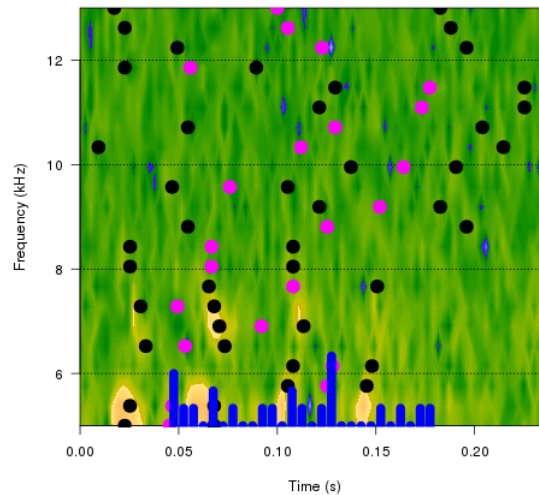


Abbildung 21: Spektrogramm eines Störgeräusches.

Beim Spektrogramm des Störgeräusches ist deutlich ersichtlich, dass die Frequenzmaxima und somit auch die Mittelpunkte keiner Struktur mehr folgen. Das Histogramm weist betreffend des Modus in Prominenz und Dominanz keine derart herausragenden Werte auf wie die beiden anderen Fälle.

Für die 14 Gruppen sollen als nächstes tabellarisch die Qualitätswerte des Monotonietests sowie des Histogrammtests dargelegt werden (siehe Tabelle 7), zusammen mit einer entsprechenden binären Bewertung. Die Zentrums-Zeit bezeichnet dabei jenen Zeitpunkt, an dem der Modus des Mittelpunktshistogramms liegt. Sie berechnet sich nicht aus den bisherigen Zeitintervallen, sondern wird hier vollständig neu bestimmt. Die Monotoniequalität liegt bei der hier benutzten Berechnungsvariante im Intervall $[-1; 1]$, die Histogrammqualität in $[0; 1]$. Bestanden ist der Monotonietest bei einem Qualitätswert innerhalb $[-0.31; +0.24]$, der Histogrammtest bei einem solchen innerhalb $[0.238; 1]$.

Nr.	Zentrums-Zeit	Monotonie-Qualität	Monotonietest	Histogramm-Qualität	Histogrammtest
1	11696	-0.25	bestanden	0.285	bestanden
2	119836	-0.35	nicht bestanden	0.095	nicht bestanden
3	138841	-0.30	bestanden	0.380	bestanden
4	139560	-0.05	bestanden	0.428	bestanden
5	140870	-0.45	nicht bestanden	0.142	nicht bestanden
6	179285	-0.25	bestanden	0.238	bestanden
7	191756	0.00	bestanden	0.190	nicht bestanden
8	221308	+0.10	bestanden	0.142	nicht bestanden
9	276737	+0.05	bestanden	0.571	bestanden
10	278021	-0.45	nicht bestanden	0.238	bestanden
11	411691	-0.10	bestanden	0.238	bestanden
12	558206	-0.20	bestanden	0.238	bestanden
13	560852	-0.50	nicht bestanden	0.238	bestanden
14	706460	-0.15	bestanden	0.333	bestanden

Tabelle 7: Resultat des Spektrogrammtests.

Fasst man alle acht Rufe, die bei beiden Tests das Prädikat „bestanden“ erhalten haben, nochmals gesondert zusammen, so ergibt sich die folgende Liste 8:

Nr.	Zentrums-Zeit	Monotonie-Qualität	Histogramm-Qualität
1	11696	-0.25	0.285
3	138841	-0.30	0.380
4	139560	-0.05	0.428
6	179285	-0.25	0.238
9	276737	+0.05	0.571
11	411691	-0.10	0.238
12	558206	-0.20	0.238
14	706460	-0.15	0.333

Tabelle 8: Rufkandidaten mit bestandenem Spektrogrammtest.

Um diese Resultate zu verifizieren, wird alternativ ein Radon-Tests durchgeführt. Der Vollständigkeit halber, und, da die beiden Testverfahren sich grundsätzlich ergänzen, werden alle 14 Rufe einem solchen Test unterzogen (siehe Tabelle 9). Die hier aufgeführten 2 Radon-Parameter bezeichnen die Standardabweichung innerhalb des definierten Winkel-Distanz-Rechtecks (siehe dazu Kapitel 3.2.4) für die linke wie für die rechte Flanke. Der Radon-Test gilt dann als bestanden, wenn die Parameter einen Wert von 2.5 überschreiten.

Nr.	Zentrums-Zeit	SA linke Flanke	Radontest links	SA rechte Flanke	Radontest rechts
1	11696	3.096	bestanden	2.587	bestanden
2	119836	1.571	nicht bestanden	1.127	nicht bestanden
3	138841	8.553	bestanden	7.843	bestanden
4	139560	9.727	bestanden	8.875	bestanden
5	140870	1.764	nicht bestanden	2.602	bestanden
6	179285	1.407	nicht bestanden	1.711	nicht bestanden
7	191756	1.917	nicht bestanden	1.979	nicht bestanden
8	221308	1.803	nicht bestanden	2.314	nicht bestanden
9	276737	13.886	bestanden	12.242	bestanden
10	278021	1.909	nicht bestanden	4.669	bestanden
11	411691	34.676	bestanden	25.767	bestanden
12	558206	14.483	bestanden	11.021	bestanden
13	560852	8.975	bestanden	4.235	bestanden
14	706460	4.379	bestanden	4.660	bestanden

Tabelle 9: Resultat des Radon-Tests.

Wie ersichtlich ist, haben 8 der obigen 14 Rufen den Radon-Test komplett bestanden. Sie seien im Folgenden zusammengefasst (siehe Tabelle 10).

Nr.	Zentrums-Zeit	SA linke Flanke	SA rechte Flanke
1	11696	3.096	2.587
3	138841	8.553	7.843
4	139560	9.727	8.875
9	276737	13.886	12.242
11	411691	34.676	25.767
12	558206	14.483	11.021
13	560852	8.975	4.235
14	706460	4.379	4.660

Tabelle 10: Rufkandidaten mit bestandenem Radontest.

Trägt man nun die Resultate beider Tests in einer gemeinsamen Tabelle 11 auf, so lassen sich durch die Bedingung, dass alle vier Teilttests gleichzeitig erfüllt sein sollen, diejenigen Rufe bestimmen, die dem Vogelkundler schlussendlich präsentiert werden.

Nr.	Zentrums-Zeit	Monotonietest	Histogrammtest	Radontest links	Radontest rechts	gesamthaft
1	11696	erfüllt	erfüllt	erfüllt	erfüllt	erfüllt
2	119836	nicht erfüllt	nicht erfüllt	nicht erfüllt	nicht erfüllt	nicht erfüllt
3	138841	erfüllt	erfüllt	erfüllt	erfüllt	erfüllt
4	139560	erfüllt	erfüllt	erfüllt	erfüllt	erfüllt
5	140870	nicht erfüllt	nicht erfüllt	nicht erfüllt	erfüllt	nicht erfüllt
6	179285	erfüllt	erfüllt	nicht erfüllt	nicht erfüllt	nicht erfüllt
7	191756	erfüllt	nicht erfüllt	nicht erfüllt	nicht erfüllt	nicht erfüllt
8	221308	erfüllt	nicht erfüllt	nicht erfüllt	nicht erfüllt	nicht erfüllt
9	276737	erfüllt	erfüllt	erfüllt	erfüllt	erfüllt
10	278021	nicht erfüllt	erfüllt	nicht erfüllt	erfüllt	nicht erfüllt
11	411691	erfüllt	erfüllt	erfüllt	erfüllt	erfüllt
12	558206	erfüllt	erfüllt	erfüllt	erfüllt	erfüllt
13	560852	nicht erfüllt	erfüllt	erfüllt	erfüllt	nicht erfüllt
14	706460	erfüllt	erfüllt	erfüllt	erfüllt	erfüllt

Tabelle 11: Zusammenstellung der Beurteilungsergebnisse.

Folgende Rufe werden dem Ornithologen nun vorgeschlagen (siehe Tabelle 12), fast, zumindest:

Nr.	Zentrums-Zeit
1	11696
3	138841
4	139560
9	276737
11	411691
12	558206
14	706460

Tabelle 12: Sieben Rufkandidaten haben alle vier Tests bestanden.

Vergleich man an dieser Stelle die anfängliche manuelle Auswahl mit dieser Liste, so fällt einem positiverweise auf, dass keine von Hand entdeckten Pfuirufe hier fehlen. Allerdings stellt der aufmerksame Beobachter auch fest, dass sich der Ruf Nr. 4 bei Sample 139560 als Geräusch anderer Art hineingeschlichen hat. Hier kommt nun der glückliche Zufall zu Hilfe, durch eine Art „Echo Cancellor“ den Ruf nachträglich ausscheiden lassen zu können. Echos sind eine häufig auftretende Nebenerscheinung bei den Aufnahmen, abgesehen von Oberwellen. Er folgt unmittelbar auf den Ruf Nr. 3, nur etwa 15 ms verzögert.

Nach dieser Behandlung durch den (menschlichen) Echo Cancellor hat sich die Liste nun um jenen Eintrag verkürzt (siehe Tabelle 13).

Nr.	Zentrums-Zeit
1	11696
3	138841
9	276737
11	411691
12	558206
14	706460

Tabelle 13: Sechs Rufkandidaten haben alle vier Tests und den Echo Cancellor bestanden.

Erfreulicherweise ist festzustellen, dass im Falle dieses Benchmarks ein relativ sehenswertes Ergebnis erfolgt ist. Dies hat die Autoren umso mehr gefreut, als sie erfuhren, dass an der Vogelwarte Sempach der Ruf Nr. 1 bisher durch rein manuelles Begutachten noch gar nie in Erscheinung getreten ist!

4 Software

Die Software dient der einfachen Anwendung der erarbeiteten Algorithmen zur Detektion und Ausmessung der Waldschnepfenbalzrufe.

4.1 Funktionen

Die Software beinhaltet folgende Funktionen:

- Import von Aufnahmedateien
- Organisieren der Aufnahmedateien in einer einfachen Struktur
- Detektieren von Rufen auf den Aufnahmedateien
- Bewertung der detektierten Rufe
- Ausmessen von Parametern aus den Rufen nach Hoodles et al.[3]
- Exportieren der Rufe und deren Parameter

4.2 Aufbau

Die Software ist in Java implementiert. Dies hat den Vorteil, dass die Software auf vielen verschiedenen Plattformen, in diesem Fall Microsoft Windows und Mac OS X, lauffähig ist. Die Algorithmen sind in der Skriptsprache R[7] implementiert. Für die Benutzeroberfläche wurde das JavaFX[9] Framework verwendet, welches Bestandteil der Java Laufzeitumgebung ist.

4.3 Ordnerstruktur

- config
- logs
- projects
 - Sammlungen
 - * Sequenzen
 - Rufe

- scripts
- whistledb
 - head
 - tail
- WoodcockScope.jar

4.3.1 Konfiguration - config

In diesem Ordner befinden sich die Konfigurationsdateien. Für das Logging sind dies die `logback.xml` und für die Software die `config.json`.

4.3.2 Logdatei - logs

Das Logback ist so konfiguriert, dass 10 Logdateien mit einer Grösse von 5 MB erstellt werden. Wenn alle 10 Dateien die 5 MB Grösse erreicht haben, werden die ältesten Dateien wieder überschrieben. In den Logdateien sind alle Ausgaben enthalten, so wie sie in `logback.xml` im config-Verzeichnis eingestellt sind.

4.3.3 Datenverzeichnis - projects

Alle Unterordner von `projects` werden als Projekte, Sammlungen, Sequenzen und Rufe in der Software angezeigt. Projekte und Sammlungen sind nur Ordner. Sequenzen sind Wave-Dateien. Die Rufe sind jeweils in einem Unterordner der Sammlung mit dem Namen der Sequenz zu finden. Die Rufe selbst haben jeweils den Namen `call_(Startsample)_(Endsample).wav`. Das dazugehörige Spektrogramm hat den selben Namen wie der Ruf und die Endung `png`. Die Radontransformierte des Rufes hat den selben Namen wie das Spektrogramm jedoch findet sich noch ein `_radon` vor der Endung. Die serialisierten Metadaten des Rufes haben auch den selben Namen wie der Ruf und haben die Endung `xml`.

4.3.4 R-Skripte - scripts

Alle R-Skripte, welche die Software verwendet, sind im Ordner `scripts` abgelegt. Änderungen an den Skripten werden nach einem Neustart der Software wirksam oder wenn im Hauptmenü unter Tools „R Umgebung neu laden“ ausgeführt wurde. Die letzte Variante

wird jedoch erst ausgeführt, nachdem der Scheduler Arbeit bekommt, da er ohne Arbeit blockiert ist. Die Neuinitialisierung wird jedoch vor dem Verarbeiten des nächsten Arbeitspaketes ausgeführt.

4.3.5 Pfuizdatenbank - whistledb

Die Pfuizdatenbank besteht aus zwei Unterordner. „Head“ für aufsteigende Flanken und „Tail“ für absteigende Flanken. Diese werden zur Vorselektion verwendet.

4.3.6 Programmdatei - WoodcockScope.jar

Die Programmdatei ist eine Jar-Datei. Jar steht für Java Archive. Diese Datei ist so konfiguriert, dass sie ausgeführt werden kann, wie die üblichen Programmdateien.

4.4 Externe Java Bibliotheken

Für das Logging wird logback verwendet und für das Speichern und Laden von Metadaten jsonio. Weiter wird für den Zugriff auf R Rserve genutzt.

4.5 JavaFX

JavaFX ist laut Oracle der nächste Schritt in der Evolution von Java als rich client platform. Es wurde entwickelt, um effiziente, Hardware-beschleunigte Java Benutzeroberflächen für kommerzielle Geschäftsanwendungen zu ermöglichen. JavaFX enthält eine grosse Menge von Steuerelementen, Grafik und Multimedia API's, um die Entwicklung von Benutzeroberflächen stark zu vereinfachen. Benutzeroberflächen können mit dem SceneBuilder per Drag & Drop zusammen gestellt werden. Im Hintergrund wird dadurch FXML[10] generiert. FXML ist eine XML-basierte Sprache, welche das Definieren von Benutzeroberflächen, unabhängig des Quellcodes für die Applikationslogik, ermöglicht. FXML ist jedoch kein Standard und wird nur bei JavaFX eingesetzt. Ähnliche Konzepte gibt es auch z.B. beim Microsoft .Net Framework mit XAML. Die Trennung von Benutzeroberflächen und Applikationslogik ist schon ein älteres Konzept [11] und wird bei JavaFX durch die deklarative Benutzeroberflächenbeschreibung sehr strikt umgesetzt. Eine Übersicht über die Architektur von JavaFX gibt die Abbildung 22

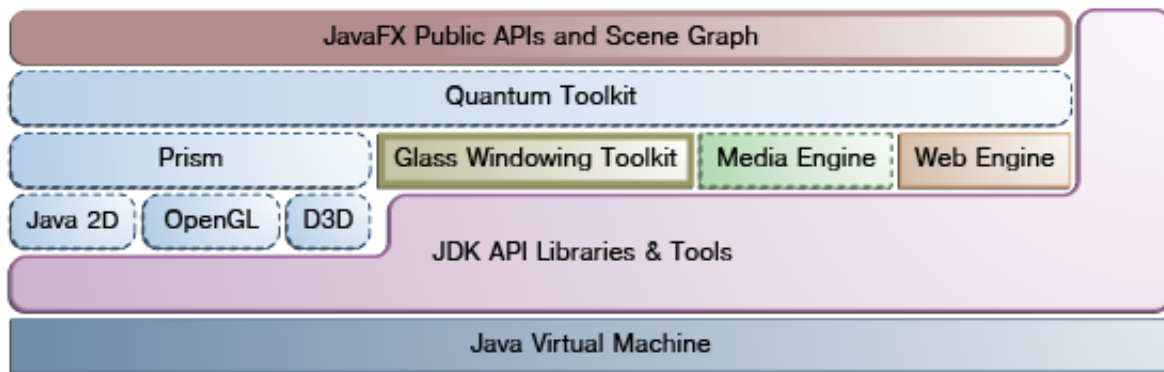


Abbildung 22: Architektur von JavaFX [12].

4.6 Lambda-Ausdrücke

Ein neues Feature von Java 8 sind Lambda-Ausdrücke. In JavaFX sind sie sehr praktisch einzusetzen.

Die alte Variante sieht folgendermassen aus:

```
btn.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        System.out.println("Hello World!");
    }
});
```

Mit Lambda-Ausdrücken sieht das gleiche Beispiel so aus:

```
btn.setOnAction((e) -> System.out.println("Hello World!"));
```

Die Lambda-Ausdrücke funktionieren immer, wenn die Schnittstelle nur eine Methode beinhaltet. Bei diesem Beispiel ist die Schnittstelle „EventHandler“ und die Methode „handle(ActionEvent event)“. Da in JavaFX darauf geachtet wurde, dass die Schnittstelle jeweils nur eine Methode beinhaltet, können für diese und ähnliche Szenarien in der Regel Lambda-Ausdrücke verwendet werden. Dadurch wird der Quellcode erheblich lesbarer und der Entwickler ist produktiver, da er die Signaturen der verschiedenen Schnittstellen nicht auswendig kennen muss.

4.7 Persistenz

Für die Persistenz von Metadaten wird jsonio verwendet. Mit jsonio können Objekte als Ganzes oder Teile davon in einem JSON-String serialisiert werden. Danach wird

dieser String in einer Textdatei abgespeichert. Beim Laden wird der Text aus der Datei abgerufen und mit jsonio zum entsprechenden Objekt deserialisiert.

Das Abspeichern von Rufmetadaten sieht z.B. wie folgt aus:

```
List<String> persistedFields = new ArrayList<String>();
persistedFields.add("startTime");
persistedFields.add("endTime");
persistedFields.add("rightSlopeQuality");
persistedFields.add("leftSlopeQuality");
persistedFields.add("monotonyQuality");
persistedFields.add("callFeatures");
```

```
File settingsFile = this.getSettingsFile();
```

```
PersistenceService persistenceService = new PersistenceService();
persistenceService.persist(this.metadata, settingsFile, persistedFields);
```

Das Laden von Rufmetadaten aus der zugehörigen Textdatei sieht dann folgendermassen:

```
File settingsFile = getSettingsFile();
if (settingsFile.exists()) {
    PersistenceService persistenceService = new PersistenceService();
    this.metadata = (CallMetadata) persistenceService.load(settingsFile);
}
```

4.8 Skriptsprache R

R ist eine Programmiersprache vom Typ Skriptsprache, die vor allem für statistische und graphische Zwecke geeignet ist. Die gesamte Umgebung (d.h. Sprache inkl. Entwicklungsumgebung) ist freie Software. R bietet eine Vielfalt von bereits implementierten Techniken und ist einfach erweiterbar. Zum Angebot gehören unter anderem lineare und nicht-lineare Modellierung, klassische statistische Tests, Zeitreihen-Analyse, Klassifikation, Clustering und vieles mehr.

Die Algorithmen wurden alle mit der Skriptsprache R entwickelt. Man kann in R effizient und flexibel Algorithmen erarbeiten, ausprobieren und verbessern. Zudem gab es schon diverse R-Pakete, die einen Grossteil der Anforderungen der Bachelorarbeit abdecken (z.B. Seewave, signal, tuneR, fftw und PET). R-Pakete sind öffentliche Bibliotheken, welche die Sprache R erweitern.

4.8.1 Verwendete Pakete

Seewave dient dem Erstellen von Spektrogrammen aus Zeitsignalen. Das Paket fftw nutzt man zum Berechnen der Korrelationen und ist eine schnellere Implementation der FFT. Aus dem PET Paket wurde die Radon-Transformation verwendet.

4.8.2 Anwendung

R-Skripte werden eingesetzt bei der

- Kreuzkorrelation der Pfuizhälften mit dem Signal
- Spektrogramm-Berechnung
- Monotonie-Analyse
- Radon-Transformation
- Auswertung der Radon-Transformation

4.9 Java zu R Schnittstelle - Rserve

Die Algorithmen, welche in R entwickelt wurden, wurden in dieser Arbeit nicht nach Java portiert, da es die einfache Lösung Rserve[13] gibt, um R-Skripte von Java aus aufzuführen.

Rserve ist ein TCP/IP Server, welcher anderen Programmen den Zugriff auf eine R-Instanz zur Verfügung stellt, ohne direkt gegen die R-Bibliothek linken zu müssen. Jede Verbindung zum Rserve-Server hat ihren eigenen Workspace und ihr eigenes Arbeitsverzeichnis. Client-seitige Implementierungen sind verfügbar für verschiedene Sprachen z.B C/C++, PHP und Java. Rserve unterstützt externe Verbindungen, Authentifizierung und Dateitransfer.

4.9.1 Java Client

Der folgende Quellcode illustriert die Einfachheit der Integration von R in Java mit Rserve:

```
RConnection c = new RConnection();  
double d[] = c.eval("rnorm(10)").asDoubles();
```

RConnection stellt die Verbindung zum Rserve-Server dar. Alle R-Befehle werden auf einem RConnection-Objekt ausgeführt.

Mit der Funktion `eval` der Klasse `RConnection` können R-Ausdrücke ausgeführt werden. `Eval` verhält sich so, wie wenn R-Ausdrücke in der R-Konsole eingegeben würden. Man kann dabei Variablen zuweisen oder abfragen. Auch der `source` Befehl, wo ganze R-Skripte geladen werden, funktioniert. Beim Zugriff auf Dateien ist zu beachten, dass die Instanz immer nur Zugriff auf den Computer hat, auf dem die Rserve Instanz ausgeführt wird.

Um grössere Datenmengen zu übertragen empfiehlt, es sich die `assign` Methode zu verwenden.

```
connection.assign("soundData", sampleBuffer);
```

Damit werden die Daten in einem binär codierten Format übertragen. Wenn es über die `eval` Methode zugewiesen wird, werden die Daten als String übertragen.

Fehlermeldungen können mit dem Befehl `geterrmessage()` abgefragt werden. Hier ist wichtig zu beachten, dass immer die letzte Fehlermeldung abgefragt wird. Falls seit dem letzten Fehler kein weiterer Fehler aufgetreten ist, wird der alte Fehler zurückgegeben. Man kann demnach nicht unterscheiden, ob es ein neuer oder alter Fehler ist. Grundsätzlich ist das nicht weiter schlimm, da die `eval` Methode eine Ausnahme (Exception) auslöst, wenn etwas schief läuft. Nach dem Auslösen der Ausnahme kann man mit `geterrmessage()` die genaue Fehlermeldung abfragen. In der ausgelösten Ausnahme selber ist die genaue Fehlermeldung von R nicht enthalten.

```
try {
    double max = connection.eval("max(spec$amp)").asDouble();
} catch (Exception e) {
    String error = connection.eval("geterrmessage()").asString();
}
```

Nachfolend weiteres Beispiel. Es beinhaltet den Aufruf der Vorselektionsroutine.

```
connection.assign("soundData", this.soundData);
connection.assign("samplingRate", new int[]{this.samplingRate});
connection.assign("sampleBits", new int[]{this.sampleBits});
```

```
String s=connection.eval(
"paste(capture.output(" +
"globalWhistles <- run.whistle.finder(soundData, samplingRate, sampleBits)" +
"),collapse='\\n')").asString();

RList whistles = connection.eval("globalWhistles").asList();
```

Zuerst werden die Variablen zugewiesen, danach die Vorselektionsroutine aufgerufen und zum Schluss wird die Ergebnisliste abgefragt. Speziell ist hier die Verwendung von `capture.output`. Diese Methode fängt alle Ausgaben über den `print` Befehl ab und gibt die Ausgaben als Ergebnis an die `eval` Methode zurück. Mit `.asString()` kann dann die gesamte Ausgabe abgefragt werden.

Leider funktioniert der `print` Befehl auf Mac OS X mit Rserve zusammen nicht korrekt. Wenn die `print` Methode nie benutzt wird, ist das kein Problem. Wird sie benutzt, müssen entweder am Schluss alle `print` Befehle entfernt werden oder es muss eine Wrapper-Methode verwendet werden, die den `print` Befehl nur auf Windows-Systemen ausführt.

4.9.2 voidEval vs. eval

Beim Ausführen von R-Code sollte man darauf achten, dass man `eval` nur benutzt, falls auch ein Ergebnis abgefragt werden möchte. Wenn es sich um einen reinen Methodenaufruf oder einen Berechnungsaufruf handelt, sollte `voidEval` verwendet werden. Für das Ergebnis macht es keinen Unterschied, jedoch kann das Verwenden der Methode `eval` zu Memory-Leaks führen.

Als Beispiel: Beim Vorberechnen der Pfuizdatenbank werden globale Variablen gesetzt, jedoch kein Ergebnis abgefragt. Der Arbeitsspeicherverbrauch des Java-Prozesses schnellte um ca. 200 MB in die Höhe, wenn `eval` verwendet wird. Dies entspricht in etwa der Grösse der Pfuizdatenbank. Dies ist unnötig, da diese Daten gar nicht abgefragt werden. Wenn die Vorberechnung der Pfuizdatenbank mit `voidEval` ausgeführt wird, ist der erhöhte Arbeitsspeicherverbrauch nicht zu beobachten.

Der Arbeitsspeicherverbrauch reduziert sich zwar nach der nächsten Garbage Collection wieder. Die Reservation des Speichers, welche auch im Task-Manager (Windows) zu sehen ist, reduziert sich jedoch nicht mehr. Dies kann dazu führen, dass die Anzeige des Arbeitsspeicherverbrauchs viel höher ist, als der tatsächliche Verbrauch. Überprüft werden kann dies mit dem Tool "Java VisualVM", welches im Java Development Kit im `bin` Verzeichnis enthalten ist.

4.9.3 Rserve-Serverinstanzen starten

Verschiedene Rserve-Serverinstanzen, welche auf dem selben Computer ausgeführt werden, unterscheiden sich grundsätzlich nur durch den TCP-Port. Beim Starten von Instanzen muss man deshalb nur den zu verwendenden TCP-Port angeben. Der folgende Quellcode-Ausschnitt illustriert, wie einfach mehrere Instanzen gestartet werden können.

```
for (int i = 0; i<numberOfInstances; i++) {  
    int port = 6311;  
  
    if (!StartRserve.checkLocalRserve(port)) {  
        logger.error("Could not start Rserve for port {}", port);  
    }  
  
    if (isWindows())  
        port += i;  
    else  
        break;  
}
```

Der erste TCP-Port ist 6311. Dies entspricht dem Standardport von Rserve. Falls beim Erzeugen von RConnection-Objekten kein Port angegeben wird, wird somit 6311 verwendet. Wenn Rserve auf Windows läuft, erhalten alle weiteren Instanzen jeweils die nächst höhere Portnummer. Auf Linux und Mac OS X ist dies nicht nötig, da Rserve pro Verbindung (RConnection) jeweils eine Kopie (fork) des eigenen Prozesses erstellt. Dieser Fork-Mechanismus ist nur auf Linux und Mac OS X verfügbar. Windows kennt dieses Konzept nicht. Darum braucht es in Windows für jede gleichzeitige Verbindung eine eigene Rserve-Serverinstanz.

Die Entwickler von Rserve stellen mit StartRserve.checkLocalRserve() die Routine für das Starten der Rserve-Serverinstanzen für Windows, Mac OS X und Linux bereits zur Verfügung. Durch minimale Anpassungen konnte die Methode für die Angabe des zu verwendenden Ports erweitert werden.

Grundsätzlich sucht diese Methode die lokale R-Installation und startet danach über den Kommandozeileninterpreter eine Rserve-Serverinstanz. Falls Standardinstallationen von R verwendet werden, funktioniert dies einwandfrei. Wenn keine Standardinstallationen verwendet werden, kann es zu Problemen führen, wenn Rserve die R Installation nicht findet. Es wird nur an gewissen Orten nach der R Installation gesucht. In Windows wird

ein Registry-Eintrag ausgelesen. Eine Möglichkeit dies zu umgehen ist, dem Benutzer zu überlassen, den R-Installationspfad anzugeben.

In der Software übernimmt der Scheduler die Aufgabe sicherzustellen, dass die Rserve-Serverinstanzen gestartet sind. Er weist auch jedem Worker-Thread eine RConnection zu. Diese RConnection behält der Worker-Thread während der gesamten Laufzeit des Programms. Wenn das Programm beendet wird, führt jeder Worker-Thread die `shutdown()` Methode auf seiner RConnection aus. Damit wird einerseits die Rserve-Serverinstanz und andererseits die Verbindung zu ihm beendet.

4.9.4 Auslastung von mehreren Kernen

Eine R-Instanz stellt eine Recheneinheit für die R-Umgebung dar. Sie besteht aus einem Prozess und einem einzelnen Thread. Ein Thread stellt einen eigenen Ausführungsstrang dar. Da Rserve nur einen Thread verwendet, kann die Ausführung per Definition nur auf einem einzelnen Prozessorkern erfolgen. Dieser einzelne Thread kann per Definition einen Multikernprozessor nicht voll auslasten. Er kann höchstens den Kern, auf dem er ausgeführt wird voll auslasten. Es braucht daher mehrere R-Instanzen, um Arbeiten parallelisieren zu können. Der Nachteil dieses Designs ist, dass dadurch nicht auf gemeinsamen Speicher zugegriffen werden kann, da die R-Instanzen jeweils einzelne Prozesse sind. Auch wenn es Möglichkeiten gibt, diese Einschränkung zu umgehen, werden sie in R nicht verwendet. Das Problem, welches sich dadurch ergibt, hat weitreichende Konsequenzen. Die Umgebung muss auf jeder R-Instanz neu geladen werden. Für diesen konkreten Anwendungsfall bedeutet das, dass die Pfuizdatenbank mehrmals geladen werden muss. Diese umfasst momentan ca. 200 MB. Dieser Speicherverbrauch multipliziert sich mit der Anzahl R-Instanzen. Die optimale Lösung wäre nur einen Prozess zu verwenden und im Prozess mehrere Threads zu starten, damit die Pfuizdatenbank nur einmal geladen werden muss. Diese ist jedoch nicht umsetzbar, weil die Architektur von R dies nicht zulässt.

Bei Rserve kommen dieselben Überlegungen zur Anwendung. Rserve stellt im Grunde nichts anderes dar als eine R-Instanz, welche über das Netzwerk erreichbar ist (siehe Abbildung 23).

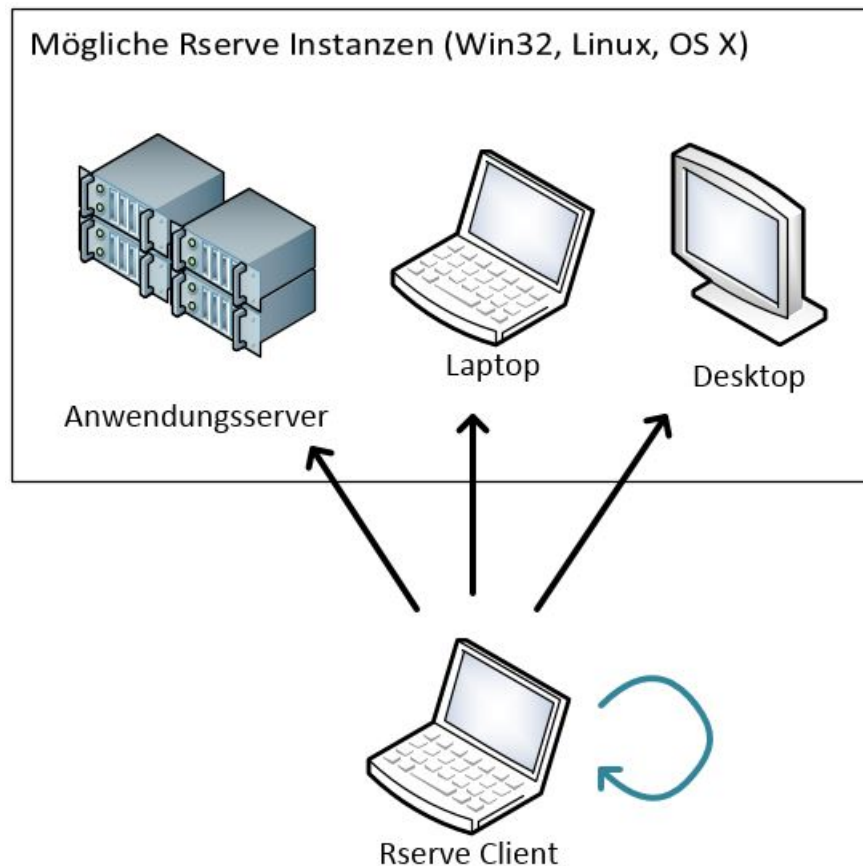


Abbildung 23: Beispiele von RServe-Instanzen.

Für die Verteilung der Arbeit auf die Rserve-Instanzen ist der Scheduler zuständig. Er besteht aus einer verlinkten blockierenden Warteschlange. Verlinkt bedeutet, dass Arbeiten nach dem FIFO-Prinzip bearbeitet werden. Blockierend bedeutet, dass der Worker-Thread, der sich Arbeit holen will, blockiert wird, wenn die Warteschlange leer ist. Dadurch wird sichergestellt, dass die Worker-Threads nur aktiv sind, wenn in der Warteschlange Arbeit vorhanden ist. Der Scheduler verteilt die Arbeit demnach nicht selbst sondern, die Worker-Threads holen sich die Arbeit, wenn welche vorhanden ist. Arbeitsobjekte müssen das Interface `IWork` implementieren.

```
public interface IWork {
    public boolean doWork(RConnection connection);
}
```

Der Worker-Thread holt sich jeweils ein `IWork`-Objekt aus dem Scheduler (siehe Abbildung 24) und ruft die `doWork`-Methode mit seiner zugewiesenen `RConnection` auf.

Dadurch wird sichergestellt, dass die RConnection nicht gleichzeitig für mehrere Arbeiten benutzt wird.

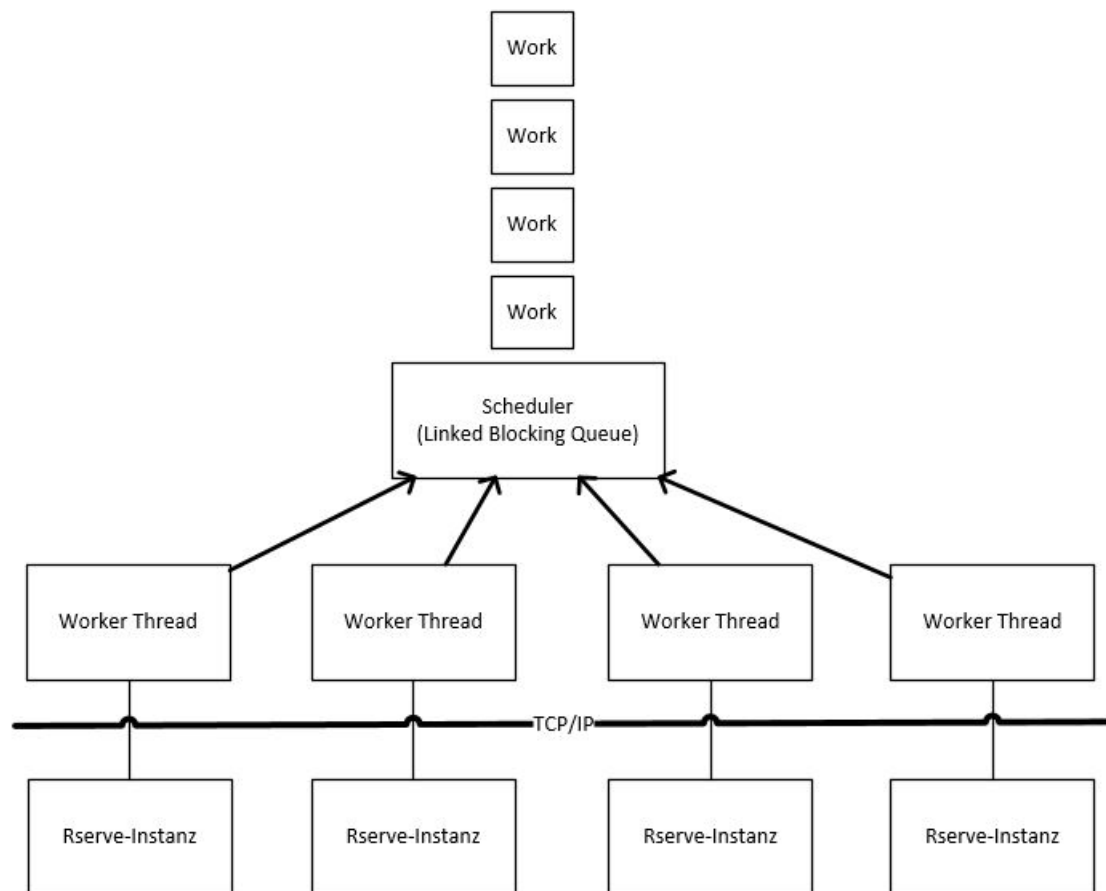


Abbildung 24: Aufbau des Schedulers.

4.9.5 Effiziente Verarbeitung von Aufnahmedateien

Die Aufnahmedateien der Vogelwarte Sempach können zum Teil eine beachtliche Grösse erreichen. Die zu verarbeitenden Dateien können von 300 MB bis mehrere GB gross sein. Deshalb muss die Verarbeitung der Dateien so konzipiert sein, dass niemals die ganze Datei in den Arbeitsspeicher geladen wird. Dafür gibt es in Java Streams. Mit Streams können Dateien sequentiell gelesen werden, wodurch nicht die ganze Datei in den Arbeitsspeicher geladen wird.

Beim Lesen und Schreiben von grossen Datenmengen sollte man im Hinterkopf behalten, dass Festplatten nicht byte- oder bitweise Daten verarbeiten, sondern immer blockweise. Diese Blockgrössen variieren je nach Hardware, Betriebssystem und Dateisystem.

Typischerweise sind es 4 KB. Es ist demnach am effizientesten jeweils 4 KB am Stück auszulesen.

Für die Verarbeitungsgrösse haben wir $2^{19} - 4096 = 520\,192$ Samples gewählt. Dies entspricht bei einer Samplingrate von 48 000 Samples pro Sekunde ca. 10 Sekunden Signal. Eine Blockgrösse wurde abgezogen, da beim Korrelieren die Signallängen der zu korrelierenden Signale addiert werden. 4 096 Samples sind für die Pfuizhälfte reserviert. Eine Pfuizhälfte darf somit aus höchstens 4 096 Samples bestehen, was 85 Millisekunden entspricht. Die Korrelation wird mit der Fast-Fourier-Transformation (FFT) gerechnet, welche bei Signallängen von Zweierpotenzen sehr effizient gerechnet werden kann. Bei den Tests in dieser Bachelorarbeit war die Zeitersparnis im Bereich Faktor 10. Durch die Wahl dieser Signallänge wird auch berücksichtigt, dass die Signallänge ein Vielfaches der Blockgrösse von 4 KB ist.

Das Arbeitsobjekt für die Voranalyse bekommt jeweils 10 Sekunden Signal zugewiesen, welches sich im Arbeitsspeicher befindet. Damit der Scheduler nicht mit Arbeitsobjekten überhäuft wird welche die Signale im Arbeitsspeicher haben, wird die Sequenzen-Analyse-Warteschlange (siehe Abbildung 25) dazwischen geschaltet. Diese ist dafür zuständig, dass der Scheduler nur jeweils eine festgelegte Anzahl Arbeitsobjekte enthält. Sobald diese Anzahl erreicht ist, werden keine weiteren Arbeitsobjekte mehr hinzugefügt. Erst wenn die Arbeitsobjekte durch die Worker-Threads ausgeführt wurden, darf wieder neue Arbeit in die Scheduler Warteschlange hinzugefügt werden. Dadurch kann der Teil der Signale, welche sich im Arbeitsspeicher befinden eingeschränkt werden.

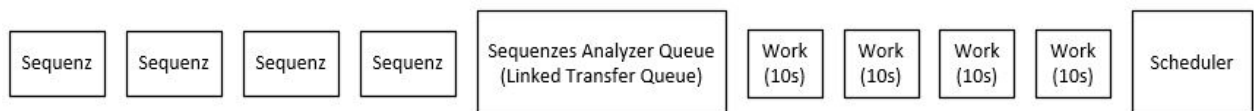


Abbildung 25: Aufbau der Sequenzen-Analyse-Warteschlange.

4.10 Benutzeroberfläche

Die Benutzeroberfläche ist in drei Arbeitsbereiche unterteilt (siehe Abbildung 26):

- Projekte
- Rufanalyse
- Analysedetails

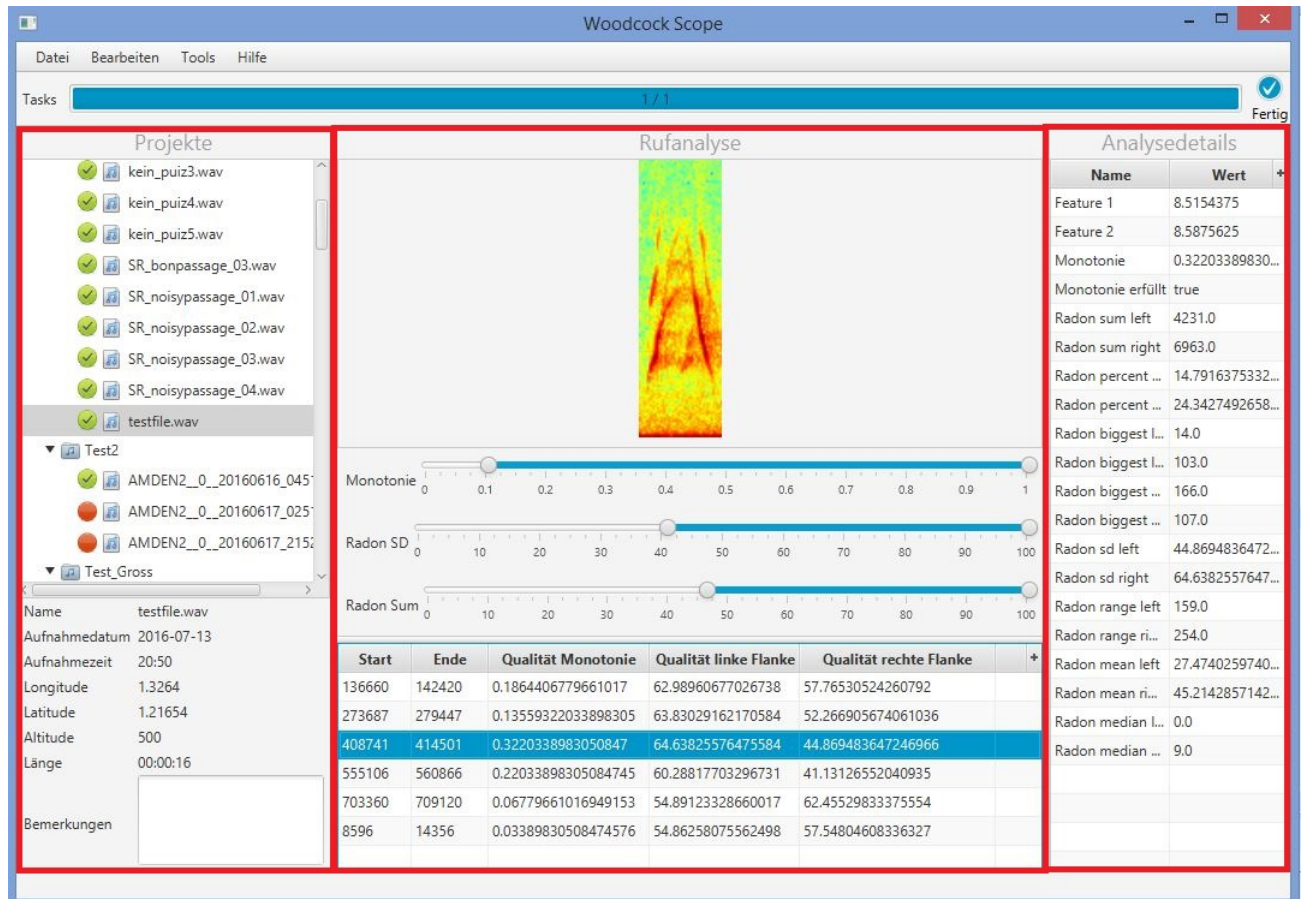


Abbildung 26: Arbeitsbereiche der Benutzeroberfläche.

4.10.1 Projekte

Im Arbeitsbereich Projekte können die Aufnahme Dateien verwaltet werden. Die Struktur lehnt sich an BatScope an. Es gibt Projekte, Sammlungen (Collections), Sequenzen (Sequences) und Rufe (Calls). Die Sequenzen zeigen an, ob sie bereits analysiert wurden. Rot bedeutet, dass sie noch nicht analysiert wurden. Grün bedeutet, dass sie bereits analysiert worden sind.

4.10.2 Rufanalyse

Im unteren Bereich der Rufanalyse wird die Liste der gefundenen Rufe und deren wichtigsten Qualitätsmasse angezeigt. Im oberen Bereich der Rufanalyse wird das Spektrogramm des ausgewählten Rufes dargestellt.

4.10.3 Analysedetails

In diesem Bereich werden zusätzliche Parameter zum ausgewählten Ruf angezeigt.

4.11 Import

Es gibt zwei Möglichkeiten Daten zu importieren, den manuellen Import und den automatischen Import aus dem Datenlogger SM3. Die Aufnahmedateien werden in das Projektverzeichnis kopiert. Das Projektverzeichnis ist ein Unterordner im Programmverzeichnis.

4.11.1 Metadaten

Zu jeder Aufnahme gibt es Metadaten. Diese sind GPS-Koordinaten, Datum, Uhrzeit, Temperatur und Bemerkungen der Aufnahmen. Diese Metadaten werden für jede Sequenz in einer gleichnamigen Datei mit unterschiedlicher Dateiendung abgespeichert. Das Format ist JSON und wird in Klartext abgespeichert.

4.11.2 Automatischer Import aus SM3

SM3 ist ein Datenlogger der Firma Wildlife Acoustics. Er wird von der Vogelwarte Sem-pach beim Monitoring der Waldschnepfe eingesetzt. Er kann in zwei Modi betrieben werden, getriggert und permanente Aufnahme. Unabhängig vom Betriebsmodus wird eine Metadatei erstellt. Diese kommaseparierte Datei enthält für jede Aufnahme-minute einen Eintrag.

4.11.3 Manueller Import

Beim manuellen Import können die zu importierenden Dateien ausgewählt werden. Es können mehrere Dateien importiert werden, ganze Ordner oder Ordner mit Unterordnern. Zusätzlich müssen die Metadaten angegeben werden. Die Metadaten werden auf die Aufnahmedateien angewendet.

4.12 Export

Die gefundenen Pfuize und die ausgemessenen Parameter nach Hoodles et al. können exportiert werden. Es wird eine kommaseparierte Datei (CSV) erzeugt. Exportiert werden Datum, Uhrzeit, verschiedene Qualitätsparameter, die Hoodles et al. Parameter, der Sequenzname und die Bemerkung zur Sequenz.

5 Appendix A - theoretische Grundlagen

In diesem Anhang wird eine kurze Übersicht über die wichtigsten mathematischen Grundlagen für das Verständnis der in der Arbeit behandelten Methoden der diskreten Signalverarbeitung gegeben. Dabei sind die folgenden Passagen mit Ausnahme des Abschnitts zur Hilbert-Transformation allesamt [4] entnommen.

5.1 Transformationen im Allgemeinen

Zu analysierende Signale (wir nehmen sie kurzerhand als reell an) liegen in aller Regel als abgetastete Zeitsignale vor, wobei jeder Abtastwert mit einem Index in den natürlichen Zahlen bezeichnet werden kann. Als Funktion betrachtet, wird jeder solche Abtastwert t_n auf seine zugehörige Signalamplitude s_n abgebildet.

$$f_1 : t_n \mapsto s_n := f_1(t_n), n \in \mathbb{N}.$$

Die im Signal enthaltene (eindeutige) Information kann jedoch je nach Anwendungsgebiet in unterschiedlicher Repräsentation gewünscht sein. Eine solche Repräsentation stellt die obige Funktion f_1 dar. Eine weitere mögliche Repräsentation ist gegeben durch die Funktion f_2 , die jedes Vielfache einer Grundfrequenz ν (hier als ν_n bezeichnet) auf den im Signal enthaltenen Anteil dieser Frequenz a_n abbildet. Dabei wird für diese Betrachtung angenommen, dass diese Anteile über das Signal hinweg konstant bleiben.

$$f_2 : \nu_n \mapsto a_n := f_2(\nu_n), \nu_n := n \cdot \nu, n \in \mathbb{N}.$$

Diese zweite Repräsentation ist eng verknüpft mit dem Resultat der reellen Fourier-Transformation eines Zeitsignals, die dem Leser noch das eine oder andere Mal begegnen wird.

Soll nun aber ein Sprachsignal analysiert werden, so lässt sich die Annahme der Konstanz dieser Frequenzanteile nicht länger halten. Sie ändern sich nämlich etliche Male innerhalb nur einer Sekunde. Die so genannte Kurzzeitanalyse baut nun darauf auf, dass Konstanz für einen Bruchteil einer Sekunde herrscht, während dessen Frequenzkomponenten bestimmt werden können und erst durch die Abfolge dieser einzelnen Spektren das Sprachsignal charakterisiert werden soll. Der Umstand, wie sich benachbarte „Signale“ beeinflussen und wie sie in Relation zu einander gewählt werden sollen, wird in 5.5 ausführlicher behandelt.

5.2 Diskrete Fourier-Transformation

Die vorliegenden Signale sind doppelt diskret, d.h. sowohl die Abtastzeitpunkte als auch die möglichen Amplitudenwerte können nur ganzzahlige Werte bezüglich einer bestimmten Basis annehmen. Sei dies z.B. der Takt des Prozessors, der für die Abtastung zuständig ist oder aber die digitale Auflösung des A/D-Wandlers, der den Wertebereich festlegt. Der massgebende Punkt hierbei ist, dass keine Signal-Information kontinuierlich verarbeitet werden muss, es sind also, anders ausgedrückt, keine Integratoren vonnöten. Daher wird das erste Verfahren, mit dem hier eine Kurzzeitanalyse betrieben wird, auch „diskrete Fourier-Transformation“, kurz DFT, genannt. Sie sei im Folgenden erläutert. Angenommen, es liegen $N \in \mathbb{N}$ Abtastwerte x_n eines (möglicherweise komplexen) Zeitsignals vor ($n < N$). Als Ausgabe liefert die DFT nun ebenfalls N Werte X_k , von denen sich jeder als Linearkombination der Signalabtastwerte $x_1, \dots, x_N$ und der komplex konjugierten Abtastwerte $e_1^H(k), \dots, e_N^H(k)$ einer diskretisierten, rein komplexen e -Funktion mit von k abhängiger Frequenz auffassen lässt.

Hinter der komplexen Konjugation (negatives Vorzeichen im Exponenten) steht der Gedanke an das Skalarprodukt zweier komplexwertiger Vektoren, genannt „Hermitesches Skalarprodukt“, bei dem einer der beiden Operanden komplex konjugiert werden muss, damit das Skalarprodukt positiv definit ist ($i \cdot (-i) = +1$).

Somit kann geschrieben werden:

$$X_k := X(k) = \sum_{n=0}^{N-1} x_n \cdot e_n^H(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-i \cdot \frac{2\pi \cdot k \cdot n}{N}} \quad (0 \leq k \leq N-1) \quad (1)$$

An dieser Stelle ist zu bemerken, dass die $e_1^{(H)}, \dots, e_N^{(H)}$ eine Orthonormalbasis (rechtwinkliges, minimales Erzeugendensystem mit Basiselements-Einheitslänge) bilden. Es wird hier aber nicht ein Vektor in kanonischer Darstellung bezüglich dieser Basis zur Multiplikation verwendet, sondern die Basiselemente werden direkt in einem Vektor zusammengefasst. Dies führt dazu, dass das Skalarprodukt eines Basiselements mit sich selbst trotz der Orthonormalität der Basis von 1 abweicht:

$$\langle e_k^{(H)}, e_k^{(H)} \rangle = N \neq 1 \quad (2)$$

Dies ist letztendlich der Grund, dass die Koeffizienten, die die DFT liefert, nicht direkt die Frequenzanteile, sondern die mit N multiplizierten darstellen. Bei der Rücktransformation muss dies wieder kompensiert werden, was durch den Vorfaktor $\frac{1}{N}$ geschieht. Eine mögliche Alternative wäre die Verwendung von $\frac{1}{\sqrt{N}}$ in beide Richtungen. Davon ab-

gesehen verläuft die Rücktransformation mit grösster Analogie zur Hintransformation:

$$x_n := x(n) = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X_k \cdot e_n(k) = \frac{1}{N} \cdot \sum_{k=0}^{N-1} X(k) \cdot e^{i \cdot \frac{2\pi \cdot k \cdot n}{N}} \quad (0 \leq n \leq N-1) \quad (3)$$

Die x_n entsprechen wie oben erwähnt den abgetasteten Signalwerten, die X_k den N -fachen Anteilen der k -ten Grundschiwingung am Signal, wobei die Periodendauer der Grundschiwingung gerade der abgetasteten Signaldauer entspricht.

Für die Betrachtung der DFT wird angenommen, dass das Zeitsignal mit der Abtastdauer für alle Zeiten periodisch fortgesetzt wird. Mehr dazu siehe in 5.5 über Fensterfunktionen. Bei der DFT können zwei essentielle Grössen variiert werden. Durch Variation der Abtastdauer eines Signals kann die Grundfrequenz verändert werden, durch Festsetzen der Anzahl Abtastpunkte (Samplingrate) kann die Anzahl verfügbarer Oberschwingungen (d.h. N) eingestellt werden.

Die DFT ist extrem tolerant. Sie liefert bei Anfrage auch Werte, die weit ausserhalb des Bereiches sind, für die gemäss des Signals eine sichere Aussage getroffen werden kann. Diese Plausibilität anhand der Parameter festzustellen, ist Sache des Anwenders. So wird nicht nur der Zeitbereich mit der Abtastzeit periodisch fortgesetzt, sondern auch das entstehende Spektrum mit der Abtastfrequenz. Ein weiteres bekanntes Phänomen, der „Alias-Effekt“, tritt auf, wenn bei einem bandbegrenzten Signal mit weniger als dem Doppelten der höchsten enthaltenen Frequenz abgetastet wird. Die Frequenz, die maximal noch detektiert werden kann, wird „Nyquist-Frequenz“ genannt.

Damit die DFT als Kurzzeitanalyseverfahren für die Spracherkennung dienen kann, wird auch hier nur ein kurzer zeitlicher Ausschnitt mit quasi-konstantem Spektrum betrachtet. Die zeitliche Veränderung der entstehenden DFT-Spektren trägt schlussendlich die Sprachinformation.

Es soll an dieser Stelle noch bemerkt sein, dass die Abtastrate, falls sie fix vorgegeben ist, geradem dem Produkt aus Zeit- und Frequenz-Auflösung entspricht. Aus dieser Tatsache ergibt sich eine Art Unschärferelation, d.h. es kann pro durchgeführter DFT nur entweder die Zeit oder die Frequenz überaus fein aufgelöst werden. Also ist immer ein situationsangepasster Kompromiss zu finden.

Die folgenden Abbildungen mögen illustrieren, wie unterschiedliche Phasen eines Balzrufs zu unterschiedlichen spektralen Darstellungen führen (siehe Abbildungen 27 und 28).

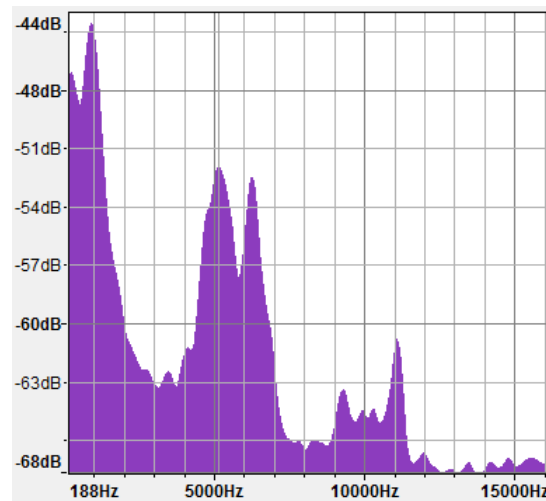


Abbildung 27: Spektrum eines ganzen Balzrufs einer Waldschnepfe.

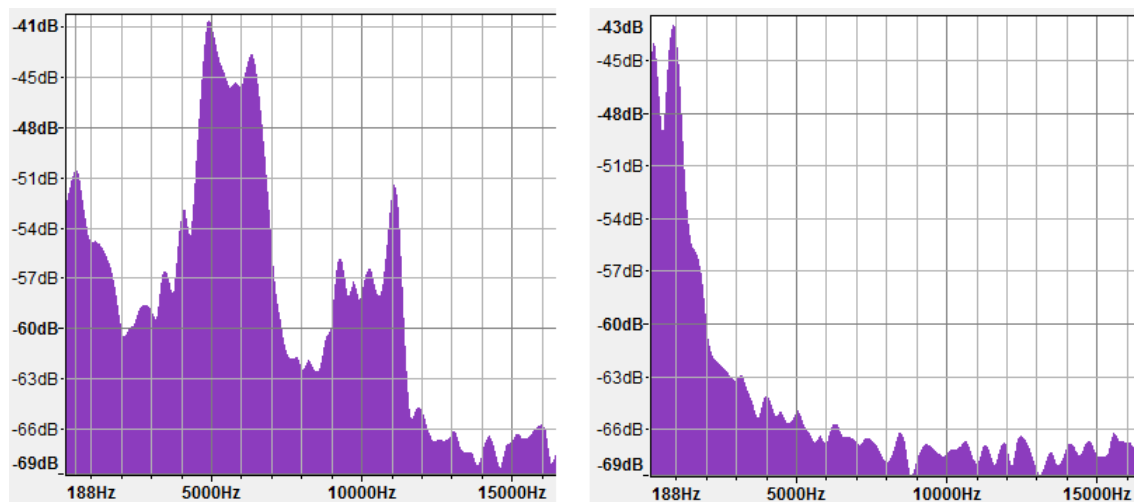


Abbildung 28: links: Spektrum des Pfuizens aus dem Balzruf einer Waldschnepfe. rechts: Spektrum eines Quorrens aus dem Balzruf einer Waldschnepfe.

5.3 Kreuzkorrelation

Die nachfolgenden Ausführungen basieren auf dem Skript [6].

In der digitalen Signalverarbeitung ist die Korrelation eine mathematische Methode zur Kombination zweier Signale, um ein drittes Signal zu erzeugen. In vielen Anwendungen zur Verarbeitung von Signalen ist es hilfreich und häufig auch notwendig, ein Mass für die Ähnlichkeit zweier Signale bestimmen zu können. Man bezeichnet dieses Mass auch

als den Grad der Korrelation, der zwischen diesen beiden Signalen vorhanden ist.

$$r_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n) y(n+k) \quad (4)$$

Die Korrelation führt zu einer Funktion $r_{xy}(k)$, welche die Übereinstimmung bzw. Ähnlichkeit eines Signals $x(t)$ zu einem zeitverschobenen Signal $y(t+k)$ in Abhängigkeit der zeitlichen Verschiebung k beschreibt. Das bedeutet, dass für jede Übereinstimmung der beiden Signale ein Maximum im Korrelationssignal auftaucht (siehe Abbildung 29).

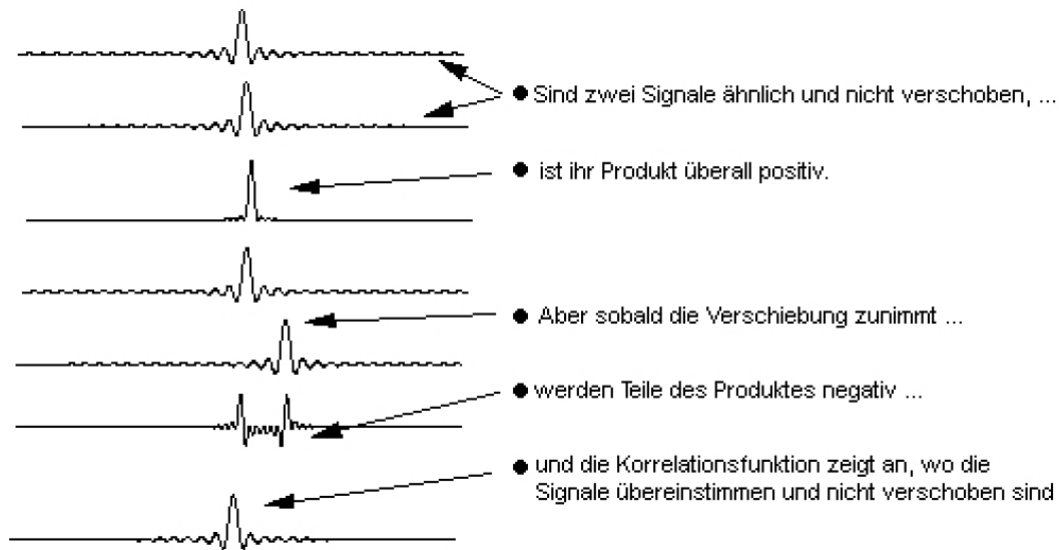


Abbildung 29: Korrelation zweier Signale [6].

Bei der Kreuzkorrelation werden zwei voneinander verschiedene Signale miteinander verglichen. Sie wird z.B. benutzt, um bekannte Referenz-Signale im Rauschen zu finden. Die Anwendung der Kreuzkorrelation auf ein Zeitsignal ist nicht immer sinnvoll. Sie kann auch auf ein transformiertes Signal wie ein Spektrogramm oder ein Cepstrum angewandt werden. Die Kreuzkorrelation wird nach Gleichung (4) berechnet.

5.4 Spektrogramme

Unter einem Spektrogramm soll in dieser Arbeit der Graph der Funktion, die jede Frequenz aus einem Signal auf ihre (aktuelle) Amplitude, resp. auf ihre (aktuelle) Phase, abbildet, verstanden werden. Optional kann auch noch eine weitere, dritte Dimension für die zeitliche Veränderung der gewählten Grösse hinzukommen. Die Darstellung des Graphen kann dabei die verschiedensten Formen annehmen. Gebräuchlich sind nebst der

zweidimensionalen Darstellung (Amplitude resp. Phase nach Frequenz) sogenannte Wasserfalldarstellungen, die eine Frequenz- und eine Zeitachse besitzen und die Intensität des jeweiligen Paares (t, ν) in Graustufen oder einer Farbskala entsprechend eintragen. Dabei wird zur Echtzeitanalyse die Zeitachse oft relativ beschriftet und der Graph laufend verschoben, wobei die neu hinzukommenden Punkte am Rande angefügt werden. Die Intensität wird bei vielen Anwendungen, vor allem aufgrund physiologischer Zusammenhänge, logarithmisch gemessen und aufgetragen (oft in der Verhältnis-Einheit dB).

5.5 Fensterfunktionen

Wie im Abschnitt 5.1 bereits erwähnt, ändert sich die Frequenzzusammensetzung des Signals laufend. Für einen kurzen Zeitpunkt jedoch kann diese als konstant angesehen werden.

Für das Ergebnis der Fourier-Transformation spielt es eine wesentliche Rolle, wie das Signal ausserhalb des betrachteten Messbereichs fortgesetzt wird. Die DFT behandelt Signale immer so, als ob gerade die gesamte Information des Signals betrachtet würde, d.h. als ob die Fundamentalperiode dem Messbereich entsprechen würde und das Signal beidseitig bis ins Unendliche fortgesetzt wäre. Unstetigkeitsstellen werden dabei als durch Delta-Distributionen verursacht angenommen, was beliebig hohe Frequenzanteile zur Folge hat. Aus diesem Grund liefert die DFT nur für bandbegrenzte Signale integere Werte. Muss nun ein aperiodisches Signal durch einen Messbereich begrenzt und transformiert werden, oder ein periodisches Signal, welches an den Rändern des Messfensters nicht den selben Wert annimmt, also keine direkte stetige Fortsetzung erlaubt, so ist vor Anwendung der DFT dafür zu sorgen, dass diese durch ihre stillschweigenden „Annahmen“ das gewünschte Ergebnis nicht verfälscht.

Zu diesem Zweck werden sogenannte „Fenster“ oder „Fensterfunktionen“ eingeführt. Ihr Hauptzweck besteht darin, den Rand des Messbereichs so zu präparieren, dass obige Bedingungen für die Bildung einer sinnvollen Fourier-Transformierten erfüllt sind. Dies kann selbstverständlich auf unterschiedliche Arten geschehen. Ideal wäre die 1-Funktion ($f(x) = 1 \quad \forall x \in \mathbb{R}$), die jedoch ein Fenster unendlicher Breite darstellt. Ihre Fourier-Transformierte entspräche einem Delta-Puls. Soll ein Fenster endlicher Breite gewählt werden, weicht auch die Fourier-Transformierte vom Delta-Puls ab. Für die Faltung der Spektren der Transformaten des Signals und des Fensters ist zur Vermeidung von Nebenlappen nötig, dass die Transformierte des Fensters eine Rechteckfunktion darstellt.

Die Funktion, die das erfüllt, ist der sogenannte „Kardinalsinus“, abgekürzt sinc.

$$\text{sinc}(x) := \frac{\sin(\pi \cdot x)}{\pi \cdot x} \quad \circ \text{---} \bullet \quad \text{rect}(\xi) := \begin{cases} 0, & |\xi| > \frac{1}{2} \\ \frac{1}{2}, & |\xi| = \frac{1}{2} \\ 1, & |\xi| < \frac{1}{2} \end{cases} \quad (5)$$

Im Folgenden sollen die zeitlich beschränkten Fensterfunktionen durch N Punkte diskretisiert werden. Die Darstellung des sinc-Fensters ergibt sich für gerade N zu:

$$\text{sinc}(n) := \begin{cases} \frac{\sin(\pi \cdot (\frac{n}{N} - \frac{N}{2}))}{\pi \cdot (\frac{n}{N} - \frac{N}{2})} & 0 \leq n < N \\ 0 & \text{sonst} \end{cases} \quad (6)$$

Häufig finden jedoch einfacher darzustellende Fenster Verwendung. Hier sollen beispielhaft drei kurz erwähnt werden, Informationen dazu lassen sich in [4] wie in [14] finden. Das in seiner Darstellung einfachste Fenster ist das Rechteckfenster (siehe Abbildung 30). Seine Fourier-Transformierte entspricht in Analogie zu oben der sinc-Funktion.

$$\text{rect}(x) \quad \circ \text{---} \bullet \quad \text{sinc}(\xi) \quad (7)$$

wobei im diskreten Fall gilt:

$$\text{rect}(n) := \begin{cases} 1 & 0 \leq n < N \\ 0 & \text{sonst} \end{cases} \quad (8)$$

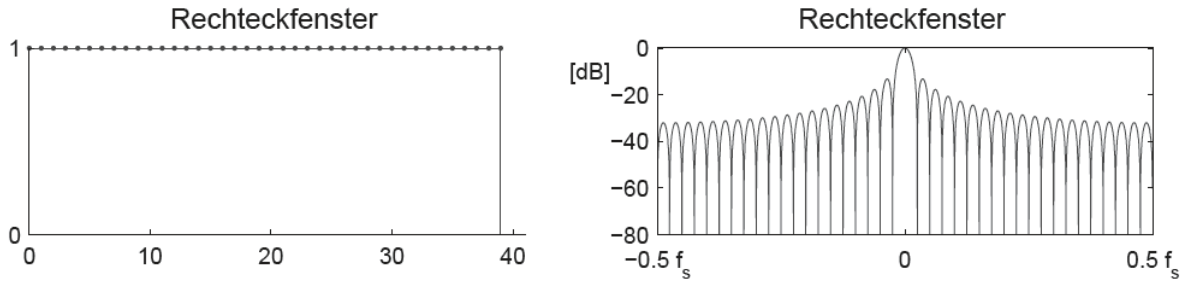


Abbildung 30: Rechteckfenster im Zeit und Frequenzbereich. [4].

Das in der Sprachverarbeitung wohl am häufigsten eingesetzte Fenster ist das sogenannte

„Hamming-Fenster“ (nach Richard W. Hamming) 31:

$$\text{hamm}(n) := \begin{cases} 0.54 - 0.46 \cdot \cos\left(\frac{2\pi \cdot n}{N-1}\right) & 0 \leq n < N \\ 0 & \text{sonst} \end{cases} \quad (9)$$

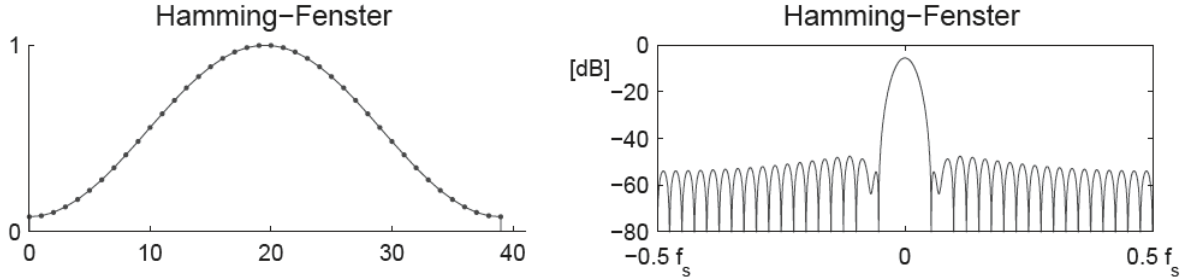


Abbildung 31: Hammingfenster im Zeit und Frequenzbereich. [4].

Als Hauptmerkmale des Hamming-Fensters kann der verglichen mit dem Rechteckfenster etwa doppelt so breite Hauptlappen gelten, zusammen mit der um 28 dB grösseren Nebenlappendämpfung. Bemerkenswert ist, dass die Werte des Signals am Fensterrand bei 0.08 liegen, statt wie üblich bei 0.

Ein weiteres gebräuchliches Fenster ist das „Hanning-Fenster“ (siehe Abbildung 32), auch einfach „Hann-Fenster“ genannt (nach Julius von Hann). Es definiert sich wie folgt:

$$\text{hann}(n) := \begin{cases} 0.5 - 0.5 \cdot \cos\left(\frac{2\pi \cdot n}{N-1}\right) & 0 \leq n < N \\ 0 & \text{sonst} \end{cases} \quad (10)$$

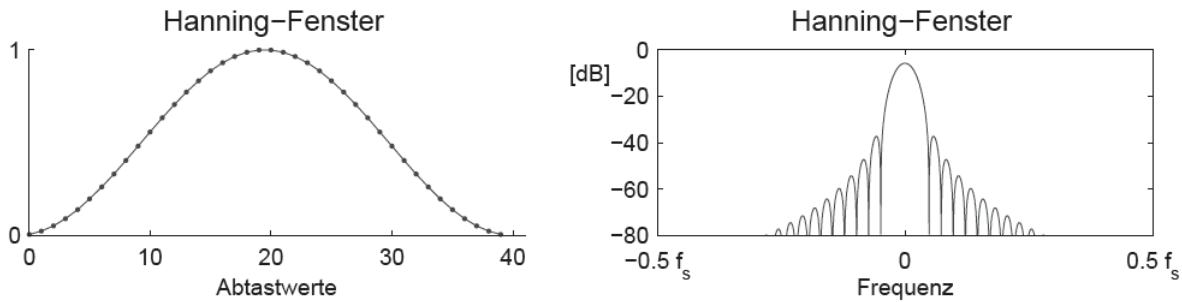


Abbildung 32: Hanningfenster im Zeit und Frequenzbereich. [4].

Es besitzt etwas steilere Flanken als das Hamming-Fenster und erreicht die 0, die Definitionen sind darüber hinaus identisch. Beim augenscheinlichen Betrachten der Fourier-

Transformierten fällt auf, dass die Nebenlappen sehr rasch abfallen und vernachlässigbare Ausmasse annehmen.

5.6 Kurzzeit-Fourier-Spektrum

In 5.2 wurde am Rande erwähnt, dass nur entweder Frequenz oder Zeit genau aufgelöst werden können. In diesem Abschnitt soll es um die genaue Zeitauflösung gehen. Das sogenannte Kurzzeit-Fourier-Spektrum ist das Resultat einer diskreten Fourier-Transformation eines sehr kurzen Signalabschnitts. Aus der Abfolge dieser Kurzzeitspektren kann, wie schon erwähnt, eine Information über das Signal gewonnen werden. Siehe dazu 5.2 und darin im Wesentlichen Abbildung 28. Eine Art, diese zeitliche Abfolge kompakt darzustellen, ist in 5.4 in Form des Wasserfalldiagramms beschrieben. Zur verwandten Thematik des „Fensters“, siehe 5.5.

5.7 Radon-Transformation

Die Radon-transformierte $r(\rho, \theta) = \mathcal{R}\{b(\mathbf{x})\}$ eines Signales $b(\mathbf{x}) = b(x_1, x_2)$ über dem Ort $\mathbf{x} = (x_1, x_2)^T$ ist definiert als [8, 2]:

$$r(\rho, \theta) = \mathcal{R}\{b(\mathbf{x})\} := \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} b(\mathbf{x}) \delta(\mathbf{x}^T \mathbf{e}_\theta - \rho) d\mathbf{x} \quad (11)$$

mit

$$\theta \in [0^\circ, 180^\circ), \quad \rho \in \mathbb{R}, \quad \mathbf{e}_\theta = \begin{pmatrix} \sin(\theta) \\ \cos(\theta) \end{pmatrix}$$

Die als Transformationskern fungierende δ -Gerade

$$\delta(\mathbf{x}^T \mathbf{e}_\theta - \rho) = \begin{cases} 0 & \text{für } \mathbf{x}^T \mathbf{e}_\theta - \rho \neq 0 \\ \infty & \text{für } \mathbf{x}^T \mathbf{e}_\theta - \rho = 0 \end{cases}$$

In der Radon-Transformation werden, für jeden Winkel θ von 0 bis 180 Grad und dem Abstand ρ vom Mittelpunkt bis zum Bildrand, die Bildwerte entlang von Geraden aufsummiert.

Abbildungsverzeichnis

1	Spektrogramm des Balzrufs einer Waldschnepfe.	4
2	Spektra deutlicher Pfuize unterschiedlicher Länge.	8
3	Andere, dem Pfuiz ähnliche, aber unsymmetrische Laute.	9
4	Pfuizrufe unterschiedlicher Scheitelfrequenzen.	9
5	Pfuizrufe unterschiedlicher Scheitelausprägungen (desselben Individuums).	10
6	Zwei vorherrschende Typen von Energieverteilungen.	10
7	<i>Aufsteigende</i> Pfuizflanke (von oben nach unten: Zeitsignal / instantane Amplitude / instantane Frequenz / instantane Phase / instantane Energie).	13
8	<i>Absteigende</i> Pfuizflanke (von oben nach unten: Zeitsignal / instantane Amplitude / instantane Frequenz / instantane Phase / instantane Energie).	13
9	Verwendeter Gauss-Filterkern.	15
10	Rechts: Korrelationsergebnisse einer steigenden Flanke (oben links) resp. einer fallenden Flanke (unten links) mit einem Pfuizruf (Mitte), in grün die ungefilterte und unverschobene Variante, in blau die nach obiger Beschreibung vorverarbeitete.	16
11	Die Monotoniebedingung ist hier relativ gut erfüllt.	21
12	Derselbe Pfuiz mit eingezeichneten Mittelpunkten zwischen den beiden Maxima je Frequenz.	22
13	Eine dritte Variante mit Histogramm der in violett dargestellten Mittelpunkte.	23
14	Spektrogramm eines Pfuiz mit Angabe des gefilterten Bereiches.	26
15	Radon-Transformierte des gefilterten Spektrogrammes aus der Abbildung oben. Die rot markierten Bereiche werden ausgewertet.	26
16	Regressionsplots der verwendeten Features. Rote Dreiecke sind Pfuize und schwarze Kreise sind keine Pfuize.	28
17	Zeitsignal und Spektrogramm der Benchmark-Aufnahme.	29
18	Zwei Pfuizrufe unterschiedlicher Qualität.	30
19	Spektrogramm des schwächsten Pfuizrufs mit Zeitmarke 11696.	34
20	Spektrogramm eines starken Pfuizrufs mit Zeitmarke 411691.	34
21	Spektrogramm eines Störgeräusches.	35
22	Architektur von JavaFX [12].	43
23	Beispiele von RServe-Instanzen.	50
24	Aufbau des Schedulers.	51

25	Aufbau der Sequenzen-Analyse-Warteschlange.	52
26	Arbeitsbereiche der Benutzeroberfläche.	53
27	Spektrum eines ganzen Balzrufs einer Waldschnepfe.	58
28	links: Spektrum des Pfuizens aus dem Balzruf einer Waldschnepfe. rechts: Spektrum eines Quorrens aus dem Balzruf einer Waldschnepfe.	58
29	Korrelation zweier Signale [6].	59
30	Rechteckfenster im Zeit und Frequenzbereich. [4].	61
31	Hammingfenster im Zeit und Frequenzbereich. [4].	62
32	Hanningfenster im Zeit und Frequenzbereich. [4].	62

Literatur

- [1] Mark D. Plumbley Dan Stowell. Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. <https://peerj.com/articles/488/>.
- [2] Blaise Mulhauser and Jean-Lou Zimmermann. Individuelle Erkennung und Bestandserfassung bei der Waldschnepfe *scolopax rusticola* anhand von Gesangsmerkmalen balzender Männchen. *Der Ornithologische Beobachter*, 2010.
- [3] Hoodless et al. Vocal individuality in the roding calls of woodcock *scolopax rusticola* and their use to validate a survey method. *Ibis*, 2008.
- [4] Beat Pfister and Tobias Kaufmann. *Sprachverarbeitung; Grundlagen und Methoden der Sprachsynthese und Spracherkennung*. Springer Verlag, 2008.
- [5] Mathias Johansson. The hilbert transform. Master's thesis, Växjö University, 1999.
- [6] Kristina Weyerhäuser. Faltung und Korrelation kontinuierlicher Signale; Seminar digitale Signalverarbeitung. <http://www.uni-koblenz.de/physik/informatik/DSV/Faltung.pdf>.
- [7] R - What is R? <https://www.r-project.org/about.html>. Accessed: 2016-07-19.
- [8] Die Radontransformation in der digitalen Bildverarbeitung. n.ethz.ch/~hitzm/day3/RadonTransformation_Anwendungsaufsatz.pdf. Accessed: 2016-07-20.
- [9] JavaFX - Developer Home. <http://www.oracle.com/technetwork/java/javase/overview/javafx-overview-2158620.html>. Accessed: 2016-07-19.
- [10] Mastering FXML. http://docs.oracle.com/javafx/2/fxml_get_started/why_use_fxml.htm. Accessed: 2016-07-19.
- [11] Separating User Interface Code. www.martinfowler.com/ieeeSoftware/separation.pdf. Accessed: 2016-07-20.
- [12] JavaFX Architecture. <http://docs.oracle.com/javafx/2/architecture/jfxpub-architecture.htm>. Accessed: 2016-07-20.
- [13] Rserve - Low-level Binary R server. <https://rforge.net/Rserve>. Accessed: 2016-07-19.

- [14] Alan V. Oppenheim, Roland W. Schaffer, and John R. Buch. *Zeitdiskrete Signalverarbeitung, 2. Auflage*. Pearson, 2004.

Bestätigung der Autorschaft

Die Autoren, Tobias Heeb und Peter Michael, bestätigen mit ihrer Unterschrift, die vorliegende Arbeit nur unter Verwendung der angegebenen Quellen und Hilfsmittel erstellt zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht und ordnungsgemäss zitiert.

Die Autoren nehmen Kenntnis davon, dass ein Plagiat als Betrug taxiert wird.

Die Arbeit hat in dieser oder in vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Ort, Datum: _____ Unterschrift: _____

Tobias Heeb

Ort, Datum: _____ Unterschrift: _____

Peter Michael