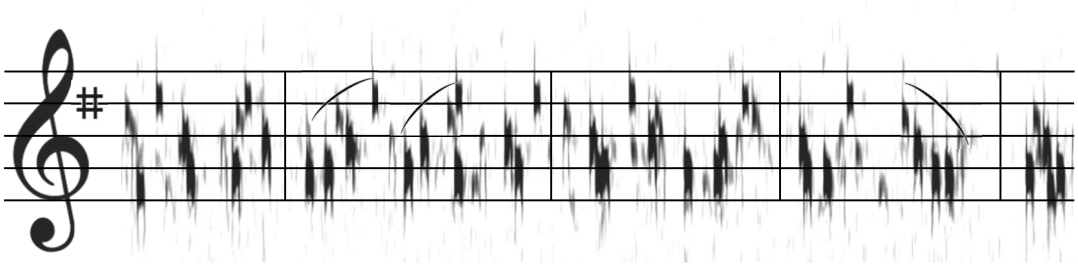
ZÜRCHER HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN
DEPARTEMENT LIFE SCIENCES UND FACILITY MANAGEMENT
INSTITUT FÜR UMWELT UND NATÜRLICHE RESSOURCEN

Akustisches Monitoring Geburtshelferkröte *(Alytes obstetricans)* – Detektor und Lokalisierung



Bachelorarbeit

von

Patrick Marti

Bachelorstudiengang Umweltingenieurwesen

Abgabedatum: 14. Januar 2021

Studienrichtung Umweltmanagement

Fachkorrektor:
Dr. Stefan Suter
Forschungsgruppe Wildtiermanagement ZHAW
Life Sciences und Facility Management, Schloss, 8820 Wädenswil

Co-Korrektorin:
Maria Jakober
Maria Jakober Umwelt GmbH
Hüslimatt 1, 6063 Stalden

Co-Korrektor:
Dr. Peter Kauf
Prognosix AG
Wohllebgasse 8, 8001 Zürich

Impressum

Titelblatt

Spektrogramm einer «Geburtshelferkröten-Symphonie» aufgenommen am 07.04.20 um 22:00 Uhr, erstellt mit RavenLite.

Zitiervorschlag

Marti P. (2021): Akustisches Monitoring Geburtshelferkröte *(Alytes obstetricans)* – Detektor und Lokalisierung. Bachelorarbeit. Zürcher Hochschule für Angewandte Wissenschaften ZHAW, Wädenswil.

Keywords

Geburtshelferkröte, *Alytes obstetricans*, Bioakustik, automatische Ruferkennung, Lokalisierung

Autor

Patrick Marti
[redacted]

Zusammenfassung

Die herkömmliche Methode, um die Bestände der stark gefährdeten Geburtshelferkröte *(Alytes obstetricans)* zu erheben, besteht darin, rufende Männchen in warmen Frühsommernächten zu zählen. Bei kleinen Populationen besteht aber die Gefahr, unregelmässige Rufer zu übersehen. Bei sehr grossen Populationen hingegen kann es schwer sein, den Bestand zuverlässig einzuschätzen. Akustische Langzeitaufnahmen bieten eine noch selten angewandte Methode, um grosse Zeiträume abzudecken und auszuwerten. Jedoch ist deren Auswertung zeitintensiv und eine Abschätzung der Anzahl Rufer nur bei kleinen Beständen möglich. Es bietet sich deshalb an, die Auswertung von Langzeitaufnahmen zu automatisieren, sowie zu testen, ob sich die Methode der Lokalisierung von Schallquellen mittels synchronisierter Aufnahmen zur Zählung von Rufern eignet.

Zur automatischen Auswertung untersuche ich die Programme Ishmael und Kaleidoscope auf ihre Eignung und entwickle eine eigene R-basierte Methode zur Ruferkennung. Die Lokalisierung teste ich in einem Laborversuch mit abgespielten Rufen, wie auch in einem Feldversuch in einer grossen Geburtshelferkröten-Population.

Dabei zeigt sich, dass das Programm Kaleidoscope nur deutliche Rufe (bis ca. 20 m vom Aufnahmegerät entfernt) erkennt, dafür aber grosse Datenmengen sehr effizient auswertet. Die R-basierte Methode findet hingegen auch leise Rufe (bis ca. 60 m vom Aufnahmegerät entfernt), braucht aber eine manuelle Nachkontrolle und die Rechendauer ist deutlich länger. Das Programm Ishmael liefert keine verwendbaren Resultate. Die Lokalisierung mittels synchronisierten Aufnahmegeräten gelingt im Laborversuch, wenn auch nicht mit sehr genauen Resultaten. Im Feldversuch scheitert die Methode hingegen. Auch wenn technische Probleme behoben werden könnten, scheint sich die Methode zur Abdeckung von grossen Flächen nur bedingt zu eignen.

Die automatische Auswertung kann in Zukunft dazu beitragen, akustische Langzeitaufnahmen effizienter auszuwerten und so deren Anwenderfreundlichkeit zu erhöhen. Dies kann dazu beitragen, kleine Populationen besser zu überwachen, sowie neue, noch unbekannte Standorte zu entdecken. Die Lokalisierung von Schallquellen sollte weiterentwickelt und auch bei anderen Arten getestet werden.

Abstract

To survey populations of the endangered Midwife Toad (*Alytes obstetricans*), it is common to count calling males during warm early summer nights. In small populations, however, there is a risk of overlooking irregular callers, while in very large populations it can be difficult to reliably estimate the population size. As a rarely used method, long-term acoustic recordings are suitable for covering and evaluating large periods of time. However, the evaluation is time-consuming and an estimation of the number of callers is only possible for small populations. Therefore, automating the evaluation of long-term recordings will be the goal. Furthermore, I will test whether the method of localizing sound sources by means of synchronized recordings is suitable for counting callers.

For automatic evaluation I tested the programs Ishmael and Kaleidoscope and developed a R-based method for call detection. I tested the localization with broadcasted calls in a laboratory experiment and in a field experiment in a large population of midwife toads.

It was shown that Kaleidoscope only detects distinct calls (up to about 20 m from the recording device) but evaluates large amounts of data very efficiently. The R-based method, on the other hand, also detects quiet calls (up to about 60 m away from the recording device) but requires a manual follow-up check and the computing time is significantly longer. The Ishmael program did not provide suitable results. In the laboratory test, localization using synchronized recorders succeeds, although with inaccurate results. In the field test, the method fails. Even if technical problems could be solved, the method seems not to be suitable for the coverage of large areas.

With automatic evaluation, long-term acoustic recordings can be evaluated more efficiently. In the future this may help to better monitor small populations, and possibly discover new sites. The localization of sound sources needs to be further developed and tested for other species.

Inhaltsverzeichnis

1 Einleitung

Die Geburtshelferkröte *(Alytes obstetricans)* ist eine kleine Amphibienart, die in der Schweiz die nördlichen Voralpen, das Mittelland und den Jura besiedelt (Meyer et al., 2014). Praktisch im ganzen Verbreitungsgebiet von der Iberischen Halbinsel über Frankreich bis nach Deutschland und in die Schweiz ist die Art rückläufig (Uthleb, 2012). In der Schweiz wird die Geburtshelferkröte als stark gefährdet (EN) eingestuft. Innerhalb von 25 Jahren sind etwa 50 % der bekannten Vorkommen erloschen (Bundesamt für Umwelt BAFU, 2005). Als Gründe werden das Verschwinden von Laichgewässern und das Fehlen von halboffenen Landlebensräumen genannt (Uthleb, 2012). An das Laichgewässer stellt die Art nur geringe Ansprüche, meist werden kleine stehende Gewässer genutzt, hingegen ist die Qualität des Landlebensraums entscheidend. Geburtshelferkröten besiedeln bevorzugt sonnige Hänge und Böschungen mit sandigen oder lockeren Böden, die nur spärliche oder gar keine Vegetation aufweisen und in der Nähe des Reproduktionsgewässers liegen (Meyer et al., 2014). Wichtig sind auch geeignete Verstecke wie Trockenmauern, Steinhaufen, Steinplatten, Holzstapel oder Mauslöcher. Die meisten grösseren Vorkommen sind in Abbaugruben wie z. B. in Kiesgruben übrig geblieben (Borgula & Zumbach, 2003).

Als einziger einheimischer Froschlurch legen die Weibchen der Geburtshelferkröte ihre Eier nicht in einem Laichgewässer ab. Das Männchen übernimmt die Brutpflege, indem es die Laichschnüre um seine Hinterbeine wickelt und die Eier erst kurz vor dem Schlupf der Kaulquappen im Laichgewässer ablegt. Der Name Geburtshelferkröte bezieht sich auf das aussergewöhnliche Verhalten bei der Übernahme der Eier durch das Männchen. Häufig wird die Kröte in Bezug zu ihrem hellen, kurzen Ruf, der an eine kleine Glocke erinnert, auch Glockenfrosch oder in der Schweiz Glögglifrosch genannt. (Uthleb, 2012)

Herkömmlicherweise werden die heimlich lebenden Geburtshelferkröten nachgewiesen, indem gezielt rufende Männchen in warmen Frühlings- oder Frühsommernächten gesucht und gezählt werden. Da einzelne Männchen oft nur unregelmässig rufen, sind mehrere Begehungen nötig (Uthleb, 2012; Bundesamt für Naturschutz BfN, 2017). Sind nur noch sehr wenige Tiere vorhanden, kann ein Nachweis sehr schwierig sein, respektive ist es gut möglich Rufer zu übersehen (Hachtel et al., 2009; Marti, 2019).

Mit akustischen Langzeitaufnahmen bietet sich heute die Möglichkeit, grosse Zeiträume abzudecken und zu analysieren (Obrist et al., 2010; Garrido Sanchis et al., 2020). Bei verschiedensten Organismengruppen gewinnen solche bioakustische Herangehensweisen an Bedeutung (Köhler et al., 2017; Sueur, 2018; Uthleb, 2020). Die automatische Auswertung mit Hilfe von Computerprogrammen spielt dabei eine wichtige Rolle (Bardeli et al., 2010; Knight et al., 2017; Burkhalter, 2019; Clink & Klinck, 2019). Im Rahmen einer Semesterarbeit habe ich die Geburtshelferkröten-Population in der Grube Mehlbach im Kanton Obwalden untersucht. Nach einem dramatischen Bestandseinbruch von mindestens 47 Rufern im Jahr 1995 auf wenige Individuen im Jahr 2009 konnten hier nach 2015 keine Geburtshelferkröten mehr nachgewiesen werden (Naturforschende Gesellschaft Obwalden und Nidwalden NAGON, 2001; Jakober, 2019). Im Frühsommer 2019 wurden während insgesamt 1890 Stunden an vier Standorten akustische Aufnahmen gemacht und manuell ausgewertet. Dabei konnte ein Rufer nachgewiesen werden. Es zeigte sich, dass sich mit dieser Methode einzelne Rufer nachwiesen lassen, die mit einem Monitoring im Feld nur schwer zu finden wären. Die manuelle Auswertung war jedoch aufwändig und eine automatische Auswertung war mit den verwendeten Methoden nicht möglich. (Marti, 2019)

Bei Ruferchören mit bis zu ca. 5 Rufern kann mit Hilfe der Langzeitaufnahmen auch die Bestandsgrösse erfasst werden. Bei grossen Ruferchören kann eine exakte Bestandserhebung sowohl im Feld, wie auch auf akustischen Aufnahmen schwierig bis unmöglich werden (Abbildung 2) (Hachtel et al., 2009). Die Bestandserfassung mit der Fang-Wiederfang-Methode und computergestützter Wiedererkennung von

Individuen anhand des Warzenmusters ist ebenfalls möglich, jedoch aufwändig und invasiv (Schlup et al., 2018). Da es aber auch bei grossen Beständen wichtig wäre, eine Bestandsabnahme frühzeitig zu erkennen um entsprechende Massnahmen zu ergreifen, wäre eine Methode zur genauen Bestandserfassung hilfreich (Hofer, 2016; Robin et al., 2017). Möglicherweise lassen sich Rufer lokalisieren und auszählen, in dem sie gleichzeitig von mehreren synchronisierten Tonaufnahmegeräten erfasst werden (Wilson et al., 2013; Rhinehart et al., 2020).

Ziel dieser Arbeit ist, einerseits die automatische Auswertung von Langzeitaufnahmen weiter zu verfolgen. Dazu werde ich zwei Programme auf ihre Eignung prüfen und falls diese den Ansprüchen nicht genügen, eine eigene Methode entwickeln. Andererseits werde ich untersuchen, ob die räumliche Verortung rufender Geburtshelferkröten mit Hilfe synchronisierter Aufnahmen möglich ist und ob sich daraus praxisorientierte Methoden ableiten lassen.

Dabei stellen sich folgende Fragestellungen:

1. Ist es möglich, Langzeitaufnahmen von Geburtshelferkröten automatisiert auszuwerten?
2. Wie gut schneidet eine automatische Auswertung im Vergleich zu einer manuellen Auswertung in Bezug zu Aufwand und Ertrag ab?
3. Ist es möglich, rufende Geburtshelferkröten mit Hilfe von synchronisierten Tonaufnahmen räumlich zu verorten und so auf die Anzahl der Rufer zu schliessen?
4. Kann diese Methode ein herkömmliches Monitoring ersetzen oder ergänzen?

2 Rufe der Geburtshelferkröte

Die Paarungsrufe der Männchen sind einfach aufgebaute, kurze und klare Laute. Die Grundfrequenz liegt zwischen 1100 und 1400 Hz (Abbildung 1). Die Dauer eines einzelnen Rufs liegt zwischen 100 und 150 ms. Die Rufrate (Anzahl Rufe pro Zeiteinheit) variiert stark: Sie liegt zwischen 4 und 100 Rufen pro Minute. Die Rufe werden aber regelmässig vorgetragen und variieren innerhalb einer Rufreihe in Bezug auf Frequenz und Rufrate kaum. Im Normalfall rufen die Männchen mit einer tiefen Rufrate, nähert sich ein Weibchen, so steigt diese stark an. Die Konkurrenz der Männchen erfolgt vor allem auf akustischem Weg. Weibchen selektieren ältere Männchen mit tiefen und langen Rufen und einer hohen Rufrate. Die Rufe der Weibchen sind ähnlich aufgebaut, aber deutlich leiser als die Rufe der Männchen. Weibchen rufen wohl, um auf sich aufmerksam zu machen und die Männchen bei der Paarung zu stimulieren. Weibchen rufen sehr selten und fast ausschliesslich alternierend mit einem Männchen während der Paarung. Langanhaltende, einzelne Rufsequenzen können also einem Männchen zugeordnet werden. Bei der Zählung von Ruferchören spielen die weiblichen Rufe kaum eine Rolle. Die Geburtshelferkröte verfügt über eine Reihe weiterer Rufe, die jedoch nur selten zu hören sind. (Uthleb, 2012)

Bei einem einzelnen Individuum konnte eine maximale Lautstärke von 84.6 dB in einer Entfernung von 15 cm ermittelt werden. Mit derselben Lautstärke abgespielte Rufe sind in 130 m Entfernung noch knapp hörbar (Marti, 2019). Häufig rufen Geburtshelferkröten aus ihren Verstecken heraus, was die Lautstärke und die Reichweite beeinflussen kann (Hachtel et al., 2009).

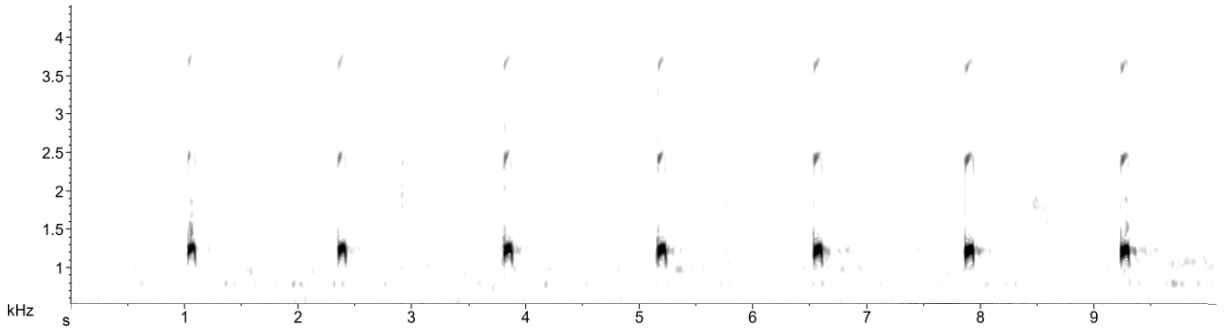


Abbildung 1: Spektrogramm einer Rufsequenz eines einzelnen Geburtshelferkröten-Männchens. Die Grundfrequenz beträgt ca. 1250 Hz, Obertöne sind bei etwa 2500 und 3750 Hz zu finden. Abbildung erstellt mit RavenLite, Quelle der Aufnahme: https://www.youtube.com/watch?v=XCU4FLUSx4I.

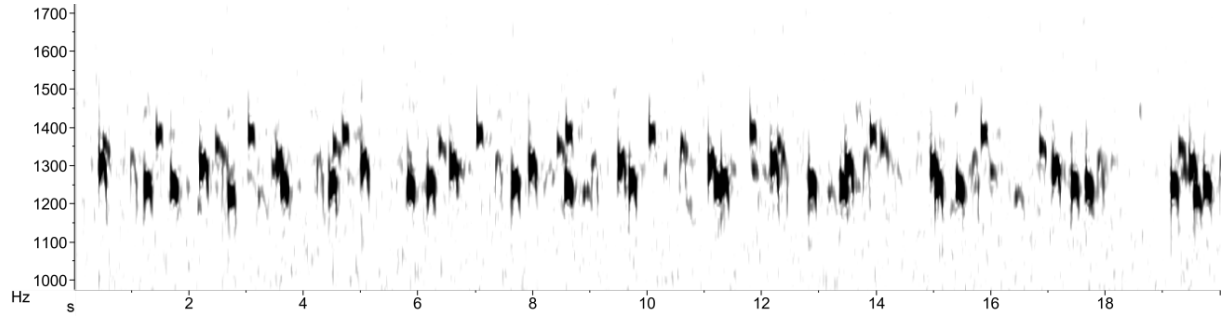


Abbildung 2: Spektrogramm von simultan rufenden Geburtshelferkröten-Männchen. Eine Abschätzung der Anzahl Rufer ist schwierig. Abbildung erstellt mit RavenLite aus eigener Aufnahme aus dem Gebiet Glaubenbielen am 07.04.20 um 22:00 Uhr.

3 Material und Methode

3.1 Verwendete Programme

Zusätzlich zu den im Folgenden genauer beschriebenen Programmen kam QGIS 3.10 zur Visualisierung räumlicher Daten zum Einsatz.

3.1.1 RavenLite

RavenLite 2.0.1 ist ein Programm zur Visualisierung und Analyse von Tonaufnahmen. Es eignet sich besonders, um Langzeitaufnahmen einfach und effizient anhand von Spektrogrammen zu analysieren. Im Programm erstellte Markierungen lassen sich als .txt-Dateien (hier Raven-Selection-Table genannt) abspeichern. Diese enthalten die Anfangs- und Endzeit sowie die minimale und maximale Frequenz einer Markierung. In R lassen sich Raven-Selection-Tables mit den Resultaten aus anderen Programmen generieren, um diese ebenfalls in RavenLite darzustellen. (Cornell Lab of Ornithology, o. J.; Köhler et al., 2017)

3.1.2 Ishmael (Pamguard)

Das Programm PAMGuard ist eine Softwareinfrastruktur zur akustischen Erkennung, Lokalisierung und Klassifizierung von Meeressäugern in Echtzeit, die auf verschiedenen Modulen aufgebaut ist. Das zur automatischen Erkennung von spezifischen Vokalisationen verwendete Modul heisst Ishmael und ist als eigenständiges Programm erhältlich (Pamguard, o. J.). Ishmael wird als benutzerfreundliche, offen zugängliche, bioakustische Analysesoftware beschrieben. Sie bietet neben der Möglichkeit zur Anzeige von Schallwellenforen und Spektrogrammen mehrere Methoden zur automatischen Ruferkennung. Es sind dies die Methoden „Energy sum detection“, „Matched filter“ und „Spectrogram correlation“ (CIMRS Bioacoustics Lab, o. J.). Da es sich in dieser Arbeit nicht um Echtzeitaufnahmen handelt, wurde nur das Programm Ishmael 3.0.2 weiterverfolgt.

Die erhaltenen Resultate speichert das Programm in einer .txt-Datei ab. Diese kann mit R in eine Raven-Selection-Table umgewandelt und in RavenLite dargestellt werden (siehe R-Skript im Anhang II).

3.1.3 Kaleidoscope Pro

Kaleidoscope Pro 5.4.0 ist ein Programm zur Darstellung, Verarbeitung und Klassifizierung von Vokalisation. Die Pro-Version verlangt eine Jahreslizenz (399 $) oder ist als Probeversion für 2 Wochen erhältlich. Dabei ist die Verarbeitung von Daten auf 4 GB beschränkt. In dieser Arbeit kam die Probeversion zur Anwendung. (Wildliefe Acoustics, o. J.)

Kaleidoscope untersucht in einem ersten Schritt Audiodateien auf Signale, die definierten Parametern entsprechen. Ein Clustering-Prozess sortiert anschliessend ähnliche Signale anhand eines mathematischen Modells und gruppiert diese in Cluster. Diese grundlegenden Cluster geben einen Überblick über die vorhandenen Vokalisationen in einem Set von Audiodateien.

Die grundlegende Clusteranalyse kann weiter durch den Aufbau einfacher oder erweiterter Klassifikatoren verfeinert werden. Ein einfacher Klassifikator basiert auf den Ergebnissen einer Clusteranalyse. Die verschiedenen Cluster werden manuell den Arten zugeteilt. Der einfache Klassifikator kann anschliessend verwendet werden, um neue Aufnahmen zu analysieren und erkannte Signale den vorher definierten Arten zuzuteilen. Diese Methode ist geeignet, um ein Artenspektrum einer Aufnahme zu erhalten.

Weiter ist es möglich, fortgeschrittene Klassifikatoren zu erstellen, die mit grossen Inputdaten manuell trainiert werden, um bestimmte Vokalisationen zu identifizieren. Ein fortgeschrittener Klassifikator ist darauf ausgelegt, so viele falsch-positive Identifizierungen wie möglich zu eliminieren. Dieser eignet sich zur Erfassung einer einzelnen Art innerhalb von Audiodateien. Er kann als Datei (.kcs-Format) abgespeichert und wiederverwendet werden.

Zur Klärung der Fragen in dieser Arbeit eignet sich demnach die Methode fortgeschrittener Klassifikator. Die erhaltenen Resultate werden in einer .csv-Datei abgespeichert. Diese kann mit R in eine Raven-Selection-Table umgewandelt und in RavenLite dargestellt werden (siehe R-Skript im Anhang II). Eine gesamtheitliche Analyse einer Audiodatei, um die erhaltenen Resultate einzuordnen und falsch-negative (nicht gefundene) Rufe zu analysieren, ist in Kaleidoscope nicht möglich.

3.1.4 RStudio (R)

Zur Datenanalyse und -verwaltung kam R (Version 3.6.1) in Kombination mit RStudio (Version 1.2.1335) zur Anwendung. R wurde ursprünglich als Programmiersprache für Statistik und grafische Darstellungen entwickelt. Heute deckt R die Bedürfnisse vieler wissenschaftlicher Disziplinen, wie zum Beispiel die der Bioakustik ab (Sueur, 2018). Spezifische Funktionen, die den Grundbedarf überschreiten werden in Paketen zusammengefasst und online zur Verfügung gestellt. Ich verwendete die Pakete „seewave“, „tuneR“ aus dem Bereich Bioakustik sowie „raster“, „sp“, „rgdal“ und „geosphere“ zur Visualisierung räumlicher Daten. Alle mit R durchgeführten Arbeiten sind mit einem R-Skript dokumentiert und somit leicht zu reproduzieren. Alle verwendeten R-Skripte befinden sich im Anhang.

3.1.5 Sound Finder

Sound Finder ist ein Programm zur Lokalisierung von Geräuschen anhand der Eintreff-Zeitdifferenzen des Schalls bei verschiedenen Mikrofonen im Raum. Das Programm ist als Excel-Tabelle oder als R-Paket, wie ich es hier verwendete, erhältlich. Als Inputdaten dient eine Tabelle mit den geografischen x-, y- und optional z-Koordinaten aller Mikrofone sowie eine Tabelle, die für jedes Geräusch die Ankunftszeit bei den jeweiligen Mikrofonen sowie die Temperatur zum Zeitpunkt der Aufnahme enthält. Das Programm berechnet zunächst die Schallgeschwindigkeit in Abhängigkeit der Temperatur zum Zeitpunkt der Aufnahme. Die Luftfeuchtigkeit wird dabei nicht berücksichtigt, da diese vernachlässigbare Auswirkungen auf die Schallgeschwindigkeit hat. Sound Finder schätzt anschliessend den Ort der Schallquelle durch Anwendung der Methode der kleinsten Quadrate (least-squares solution), die für globale Positionierungssysteme entwickelt wurde. Falls eine Höhenkoordinate der Mikrofone vorhanden ist, werden Geräusche automatisch in den drei Dimensionen lokalisiert, ansonsten nur in zwei. Für eine dreidimensionale Loka-

lisierung sind mindestens vier Mikrofone im Raum nötig, für eine zweidimensionale Lokalisierung mindestens drei. Als Output erzeugt Sound Finder eine Tabelle, die x-, y- und allenfalls z-Koordinaten der Schallquelle und eine Schätzung des Fehlers, der mit der Lokalisierung verbunden ist, enthält. Höhere Fehlerwerte (in Meter angegeben) bedeuten ein geringeres Vertrauen in die Genauigkeit der Lokalisierung. (Wilson et al., 2013)

3.2 Verwendete Tonaufnahmen zur automatischen Auswertung

3.2.1 Übungsfile

Um die Detektoren in unterschiedlichen Situationen zu testen und unter sich zu vergleichen, erstellte ich ein Übungsfile von ca. acht Minuten Länge (Abbildung 3). Dasselbe File verwendete ich bereits in der vorausgehenden Semesterarbeit (Marti, 2019). Der Abschnitt 1 enthält normales Hintergrundrauschen ohne Rufe. Im Abschnitt 2 finden sich 22 Rufe einer Rufsequenz eines einzelnen Individuums. Abschnitt 3 besteht aus Rufsequenzen von mehreren simultan rufenden Individuen. Der vierte und letzte Abschnitt enthält starke Störgeräusche von Kuhglocken.

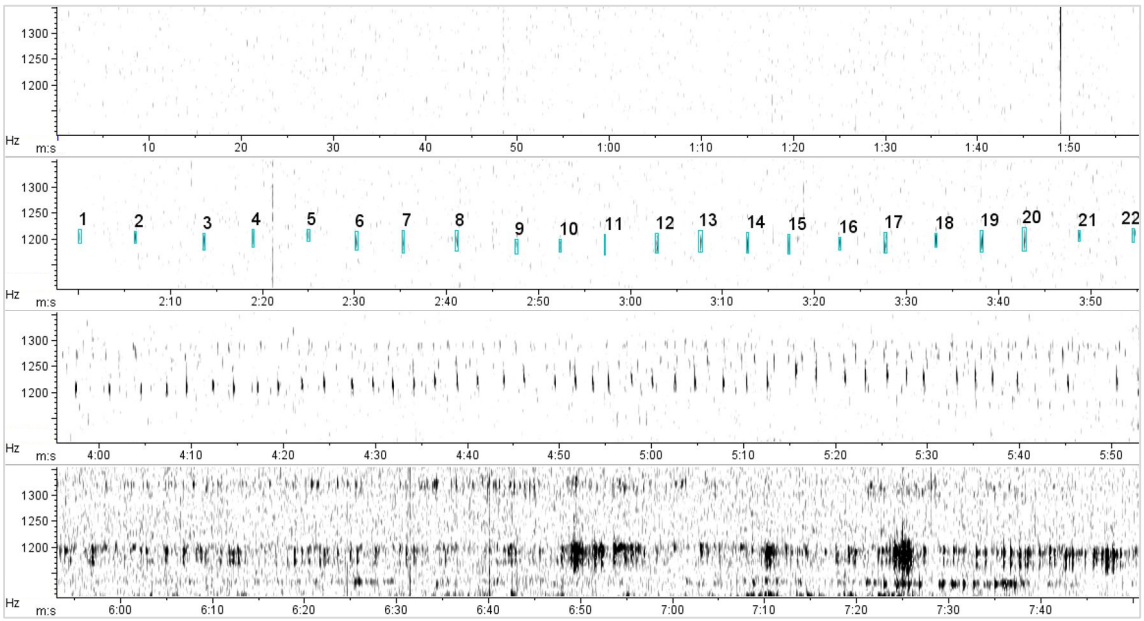


Abbildung 3: Spektrogramm des Übungsfiles. Von oben nach unten Abschnitt 1-4. Blau markiert die 22 Rufe in Abschnitt 2. Abbildung erstellt mit RavenLite.

3.2.2 Langzeitaufnahmen aus der Mehlbachgrube

In der Grube Mehlbach im Kanton Obwalden wurden 2019 an vier Standorten Langzeitaufnahmen gemacht, um zu untersuchen, ob sich im Gebiet noch Rufer befinden. Ein Distanztest half dabei, abzuschätzen bis zu welcher Distanz die Aufnahmegeräte Rufe noch aufnehmen und sich diese in einem Spektrogramm darstellen lassen. An einem Standort (Gerät OW04) wurden dazu mit einem Lautsprecher abgespielte Rufsequenzen in 5, 10, 20, 50, 60, 80 und 130 m Distanz aufgenommen. Die Lautstärke der Rufsequenz war dem gemessenen Wert von 84.6 dB in 15 cm Entfernung zu einem rufenden Geburtshelferkrötenmännchen angepasst.

Die Langzeitaufnahmen wurden 2019 mit dem Programm RavenLite manuell ausgewertet. Der Aufwand zur Auswertung betrug 30.4 h (14.4 h für die Aufnahmen der Geräte OW03 und OW04). Bei einem Standort (Gerät OW03) fanden sich in 18 Nächten Rufsequenzen eines einzelnen Rufers mit einer Gesamtdauer von 11.1 h. Die Rufe dieser Aufnahme waren teilweise sehr leise. Anhand der Ergebnisse des Distanztests wurde die Entfernung des Rufers zum Aufnahmegerät auf ca. 20-50 m geschätzt. Die beiden anderen Geräte (OW01 und OW02) enthielten keine Rufe und werden deshalb hier nicht verwendet. (Marti, 2019)

Tabelle 1: Einstellungen der SM3 Aufnahmegeräte.

Einstellungen	
Samplingrate (sample rate)	8 kHz
Samplingtiefe (sample size)	16 Bit
Aufnahmezeit	20:00 bis 06:00 Uhr
Dauer pro File	5 h
Gesamt Dauer	1890 h

3.2.3 Langzeitaufnahmen aus der Dickbangrube

Um auch Aufnahmen mit deutlicheren Rufen zu analysieren, verwendete ich zusätzlich Langzeitaufnahmen aus der Dickbangrube im Kanton Solothurn aus dem Jahr 2017. Diese stellte Esther Schweizer, Regionalvertreterin der karch, zur Verfügung. Die Dickbangrube beherbergt ein kleines Vorkommen von ca. fünf Rufern (pers. Mitteilung E. Schweizer). Diese sind deutlich näher beim Aufnahmegerät. Somit sind die Aufnahmen lauter als die aus der Mehlbachgrube. Diese Aufnahmen aus der Dickbangrube wurden bisher noch nicht systematisch ausgewertet. Zum Vergleich mit den automatischen Auswertungen habe ich diese Aufnahmen manuell nach derselben Methode wie die Aufnahmen der Mehlbachgrube ausgewertet.

Tabelle 2: Einstellungen der Swift Aufnahmegeräte.

Einstellungen	
Samplingrate (sample rate)	24 kHz
Samplingtiefe (sample size)	16 Bit
Aufnahmezeit	Sonnenuntergang bis Sonnenuntergang + 3 h
Dauer pro File	3 h
Gesamt Dauer	69 h

3.3 Automatische Auswertung

Ziel der automatischen Auswertung ist es, von einem Programm Vorschläge zu erhalten, wo sich in der Aufnahme Rufe befinden. Diese Vorschläge sollen anschliessend manuell überprüft werden. Diese Überprüfung muss mit deutlich weniger Aufwand verbunden sein als eine komplette manuelle Auswertung.

In einem ersten Schritt wendete ich die Detektoren am Übungsfile an. Um die Leistung der Detektoren unter sich und mit den tatsächlichen Gegebenheiten (Ground truth) zu vergleichen, erstellte ich eine Vergleichstabelle. Für die Abschnitte 1 und 4 des Übungsfiles (Umgebungsrauschen resp. Störgeräusche) sind die Anzahl detektierter Signale angegeben. Im besten Fall liegen diese bei null. Im Abschnitt 2 (einzelne Rufsequenz) verglich ich manuell und automatisch markierte Rufe. Dies ergibt eine Wahrheitsmatrix (Tabelle 3) (Sueur, 2018). In der Vergleichstabelle ist die Anzahl der einzelnen Positionen in der Wahrheitsmatrix (TP, FP, FN) angegeben. Die Position TN wird nicht behandelt.

Tabelle 3: Wahrheitsmatrix zum Vergleich der automatisch generierten Resultate mit den tatsächlichen Gegebenheiten. Die Abkürzungen stehen für true positive (TP), false positive (FP), false negative (FN) und true negative (TN).

Detektor	manuell (Ground truth)	
	positiv (Ruf)	negativ (kein Ruf)
positiv (Markierung)	Richtig Positive (TP): Rufe, die als solche markiert sind	Falsch Positive (FP): Markierungen, die keinen Rufe enthalten
negativ (keine Markierung)	Falsch Negative (FN): Rufe, die nicht markiert sind	Richtig Negative (TN): Abschnitt ohne Ruf wird nicht markiert

Der Abschnitt 3 eignet sich für eine systematische Auswertung nicht, da sehr viele Rufe vorhanden sind und es schwierig ist, klar zu definieren, was ein Detektor als Ruf erkennen soll und was nicht. Der Detek-

tor soll hier über den ganzen Abschnitt Rufe erkennen. Die Beurteilung erfolgt über Kommentare. Weiter enthält die Tabelle Informationen zur Rechendauer sowie Angaben dazu, ob der Output Informationen zu Rufdauer und Ruffrequenz enthält. Eine Methode verfolgte ich jeweils nur weiter, falls sie die Kriterien in Tabelle 4 in der Anwendung am Übungsfile erfüllte.

Tabelle 4: Kriterien zur Beurteilung der Resultate des Übungsfiles.

Abschnitt	Kriterien
1	1.) weniger als 5 falsch Positive
2 und/oder 3	2.) während mehr als 80 % der Zeit richtig Positive
4	3.) falsch Positive auf weniger als 30 % der Zeit verteilt
	4.) Parameter sind universal und lassen sich auf andere Aufnahmen übertragen
	5.) vertretbare Rechenzeit

Die Filtermethode und das Programm Kaleidoscope erfüllten diese Kriterien. Mit beiden Detektoren wertete ich anschliessend alle Aufnahmen der Geräte OW03 (leise Rufe) und OW04 (inkl. Distanztest) aus der Mehlbachgrube sowie die Aufnahmen aus der Dickbangrube (laute Rufe) aus.

Um die Resultate aus den Langzeitaufnahmen unter sich und mit den Resultaten der manuellen Auswertung zu vergleichen, erstellte ich wiederum eine Vergleichstabelle, die alle vier Positionen aus der Wahrheitsmatrix (TP, FP, FN, TN) enthält. Zudem wird die jeweilige Sensitivität und Spezifität der Detektoren angegeben. Diese berechnen sich folgendermassen (Lalkhen & McCluskey, 2008):

$$Sensitivität = \frac{TP}{TP + FN} * 100$$

$$Spezifität = \frac{TN}{TN + FP} * 100$$

Die Sensitivität beschreibt, wieviel Prozent der Zeit mit tatsächlichen Rufreihen der Detektor korrekt erkannte. Die Spezifität beschreibt, wieviel Prozent der Zeit ohne Rufreihen der Detektor korrekt erkannte. Die Daten wurden mittels des R-Skripts in Anhang III berechnet. Die Ergebnisse visualisierte ich in einer Linienbereichsdarstellung (R-Skript in Anhang IV).

3.3.1 Ishmael

Anhand des Tutorials (CIMRS Bioacoustics Lab, o. J.) und den mitgelieferten Audiodateien liessen sich die Methoden „Energy sum detection“, „Spectrogram correlation“ und „Matched filter detector“ umsetzten und die beschriebenen Resultate replizieren. Anschliessend passte ich die entsprechenden Parameter dem Übungsfile an. Da die Resultate des Übungsfiles den gestellten Kriterien nicht genügten, wertete ich mit Ishmael keine Langzeitaufnahmen aus.

Die am einfachsten aufgebaute Methode ist „Energy sum detection“. Dabei wird der Schallpegel in einem bestimmten Frequenzband des Spektrogramms gemessen. Als Parameter werden die obere und untere Grenze des gewünschten Frequenzbands sowie ein Schwellenwert, ab dem ein Signal detektiert wird, angegeben. Die auf das Übungsfile abgestimmten Parameter sind in Tabelle 5 ersichtlich.

Tabelle 5: Parametereinstellungen Ishmael „Energy sum detection“.

Parameter	Einstellung	Bemerkung
Lower frequency bound	1000 Hz	Untergrenze des Frequenzbands
Upper frequency bound	1500 Hz	Obergrenze des Frequenzbands
Detection threshold	2.6	Schwellenwert, wird anhand des Ergebnisses festgelegt
Minimum duration	0 s	Mindestdauer der Vokalisation
Maximum duration	1 s	Maximaldauer der Vokalisation
Detection neighbourhood	1 s	Mindestabstand zur nächsten Vokalisation

Die Methode „Spectrogram correlation" ist eine Erkennungsmethode, die sich für uniforme Vokalisationen eignet, die mit einfachen Frequenzkonturen umrissen werden können. Dazu werden die Dauer (in Start- und Endzeit) sowie die Start- und Endfrequenz der Vokalisation und eine Konturbreite angegeben. Start- und Endfrequenz unterscheiden sich nur bei an- oder absteigenden Vokalisationen. Die auf das Übungsfile abgestimmten Parameter sind in Tabelle 6 ersichtlich.

Tabelle 6: Parametereinstellungen Ishmael „Spectrogram correlation".

Parameter	Einstellung	Bemerkung
Contour width	5 Hz	Konturbreite
Start time	0 s	Startzeit entspricht bei einsilbigen Vokalisationen 0 s
End time	0.5 s	Endzeit entspricht der Dauer der Vokalisation
Start freq	1210 Hz	bei gleichbleibenden Vokalisationen erhalten Start- und Endfre-
End freq	1210 Hz	quenz denselben Wert
Detection threshold	51	Schwellenwert, wird anhand des Ergebnisses festgelegt
Minimum duration	0 s	Mindestdauer der Vokalisation
Maximum duration	1 s	Maximaldauer der Vokalisation
Detection neighbourhood	1 s	Mindestabstand zur nächsten Vokalisation

Die Methode „Matched filter" arbeitet, indem sie das Eingangssignal mit einem anderen Signal, dem Kernel, kreuzkorreliert. Als Kernel verwendete ich einen 1.5 s langen Ausschnitt aus dem 2. Abschnitt des Übungsfiles, der einen Einzelruf enthält (Tabelle 7).

Tabelle 7: Parametereinstellungen Ishmael „Matched filter".

Parameter	Einstellung	Bemerkung
Detection threshold	0.018	Schwellenwert, wird anhand des Ergebnisses festgelegt
Minimum duration	0 s	Mindestdauer der Vokalisation
Maximum duration	1 s	Maximaldauer der Vokalisation
Detection neighbourhood	1 s	Mindestabstand zur nächsten Vokalisation

3.3.2 Kaleidoscope

Zum Manual des Programms Kaleidoscope (Wildliefe Acoustics, o. J.) werden keine Audiodateien mitgeliefert. Deshalb arbeitete ich sogleich mit den originalen Daten. In einem ersten Schritt prüfte ich, ob die grundlegende Clusteranalyse die Rufe im Übungsfile findet, dies noch ohne die Signale zu klassifizieren. Die dazu verwendeten Parametereinstellungen sind in Tabelle 8 ersichtlich.

Tabelle 8: Parametereinstellungen Kaleidoscope „grundlegende Clusteranalyse".

Parameter	Einstellung	Bemerkung
Signal Params		
Minimum Frequency Range	1100 Hz	Untergrenze des Frequenzbands
Maximum Frequency Range	1400 Hz	Obergrenze des Frequenzbands
Minimum Length of Detection	0 s	Mindestdauer der Vokalisation
Maximum Length of Detection	0.5 s	Maximaldauer der Vokalisation
Maximum inter-syllable gap	1 s	Mindestabstand zur nächsten Vokalisation
Cluster Analysis		
Methode	Scan recordings and extract detections (no clustering)	
FFT Window	21.33 ms	Fensterlänge zur Fourier Transformation

Zur weiteren Verfeinerung der Resultate erstellte ich anschliessend ein fortgeschrittener Klassifikator. Dieser Prozess verlangt eine ausreichend grosse Menge an Eingabedaten. Diese werden zunächst mit einer grundlegenden Clusteranalyse bearbeitet. Da sich bereits bei der Analyse des Übungsfiles zeigte, dass leise Rufe, wie sie in den Aufnahmen der Mehlbachgrube zu finden sind, nicht erkannt werden, verwendete ich eine dreistündige Aufnahme aus der Dickbangrube sowie eine einstündige Aufnahme aus dem Gebiet

Glaubenbielen (siehe Kap. 3.4.3). Die Einstellungen im Abschnitt „Signal Params“ entsprechen der Tabelle 8, für den Abschnitt „Cluster Analysis“ der Tabelle 9.

Tabelle 9: Parametereinstellungen Kaleidoscope „grundlegende Clusteranalyse“.

Parameter	Einstellung	Bemerkung
Cluster Analysis		
Methode	Scan and cluster recordings to create cluster.kcs and cluster.csv	
Max distance from Cluster centre to include in cluster.csv	2	Maximum, alle Signale werden einbezogen
FFT Window	21.33 ms	Fensterlänge zur Fourier Transformation
Max states	12	Grundeinstellung
Max distance to cluster centre for building clusters	0.5	Grundeinstellung
Max clusters	500	Maximale Anzahl Cluster

Die dabei gefunden Signale (2125) habe ich anschliessend manuell als Ruf (1521), andere Geräusche (467) oder unsichere bzw. Rufe mit Überlagerung durch andere Geräusche (137) eingeteilt. Im Programm wird dies folgendermassen implementiert: In die Spalte „Manual ID“ der Resultate-Tabelle wird bei einem Ruf „ALOB“ (Abk. *Alytes obstetricans*) eingetragen, bei einem unsicheren oder überlagerten Ruf bleibt die Spalte leer und bei anderen Geräuschen bleibt die automatisch vergebene Bezeichnung des Clusters bestehen.

Im nächsten Schritt werden dieselben Eingabedateien erneut analysiert, diesmal jedoch mit der Methode „Re-scan recordings and edited cluster.csv to create new cluster.kcs with pairwise classifiers and cluster.csv” (weitere Parametereinstellungen wie in Tabelle 8 und 9). Die dabei entstehende .kcs-Datei kann nun als Klassifikator für weitere Audiodateien verwendet werden.

Zum Testen des Klassifikators habe ich zwei Audiodateien derselben Aufnahmen, wie sie für die Erstellung des Klassifikators verwendet wurden, ausgewertet. Die dabei verwendeten Parametereinstellungen entsprechen jenen der Tabellen 8 und 9, als Methode wählte ich „Use existing .kcs to sort new recordings and create new cluster.csv“.

Bei den Resultaten untersuchte ich, wie viele davon falsch positiv sind (3,3 %). Falls die Resultate des Klassifikators nicht den Ansprüchen genügen, kann der Prozess mit zusätzlichen Daten wiederholt werden.

Um die Resultate aus den Langzeitaufnahmen mit anderen Resultaten und der manuellen Auswertung zu vergleichen, habe ich die erkannten Einzelrufe mittels R-Skript (Anhang V) zu Rufreihen zusammengeführt. Als Kriterien, damit Rufe in einer Rufreihe zusammengefasst werden, definierte ich über den maximalen Intervall von 40 s zwischen zwei Rufen einer minimalen Anzahl von 6 Rufen pro Rufreihe (einzelne Rufe, die diesen Kriterien nicht entsprechen sind häufig falsch Positive).

3.3.3 Filtermethode

Die Filtermethode ist eine selbstentwickelte R-Funktion mit der Bezeichnung „find.alob“. Die Methode hat das Ziel, Rufreihen der Geburtshelferkröte anhand der Regelmässigkeit der Rufe automatisch zu erkennen (R-Skript im Anhang VI). Dabei filtert das Programm in einem ersten Schritt alle Geräusche aus der Aufnahme heraus, die den Kriterien eines Einzelrufs entsprechen. Anschliessend werden Einzelrufe herausgefiltert, die zu einer regelmässigen Rufreihe zusammengeführt werden können. Im Anhang XIII findet sich eine detaillierte Anleitung zum Gebrauch der Funktion. Weiter entwickelte ich drei Funktionen zur Handhabung der Resultate (Tabelle 10).

Tabelle 10: R-Funktionen zur Erkennung von Geburtshelferkröten-Rufen sowie zur Handhabung der Resultate.

Funktion	Beschreibung	Anhang
find.alob	sucht Rufreihen von Geburtshelferkröten in einem Set von Audiodateien	VI
subset.output	entfernt falsch positive Rufsequenzen nach der manuellen Nachkontrolle aus dem Output	VII
join.output	fügt die einzelnen Output-Tabellen zu einer Tabelle zusammen und berechnet Start- und Endzeit*	VIII
create.soundfiles	speichert die gefundenen Rufsequenzen als einzelne Audiodateien ab	IX

** Audiodateien müssen dazu folgendermassen benannt sein: [Gerätebezeichnung]_JJJJMMTT_hhmmss.wav*

Die Funktion „find.alob“ ist in 8 Einzelschritte unterteilt. Die darin erwähnten *Parameter* sind in Tabelle 11 erläutert.

1. Einlesen der Audiodatei

Die zu analysierende Audiodatei wird jeweils in Abschnitten von 2 min *(section.length.min)* eingelesen und zur schnelleren Verarbeitung auf 8000 Hz heruntergetaktet. Der Audioinputkanal wird mit *channel* ausgewählt.

2. Fourier Transformation

Mit der Funktion *spectro* wird eine Kurzzeit-Fourier-Transformation mit einer Fensterlänge von 512 Hz durchgeführt. Aus der ursprünglichen Wellenform der Audiodatei wird dadurch eine Matrix (Spektrogramm). Diese ist in x-Richtung in Zeitfenster mit der Länge von 0.064 s und in y-Richtung in Frequenzfester von 15.625 Hz unterteilt. Jede Zelle, sogenannte Heisenbergbox, enthält einen Fourier-Koeffizienten, der der Amplitude entspricht und im Spektrogramm in der z-Achse (farbliche Abstufung) dargestellt wird. Die Matrix wird zur schnelleren Verarbeitung auf den Frequenzbereich von 1000 bis 1500 Hz beschränkt und enthält noch 33 Frequenzfenster (Zeilen) und 1875 Zeitfenster (Spalten). (Sueur, 2018)

3. Definieren des Schwellenwerts

Um das gewünschte Signal zu finden, wird ein Schwellenwert definiert, der Signal und Hintergrundrauschen unterscheidet. Dazu wird der Median der Matrix errechnet und mit der doppelten Standardabweichung sowie einem zusätzlichen Faktor *(threshold.factor)* multipliziert. Die Standardabweichung gibt dabei an, wie stark das Hintergrundrauschen ist. Der zusätzliche Faktor gibt vor, wie stark sich ein Signal vom Hintergrundrauschen abheben muss, um erkannt zu werden. Alle Zellen der Matrix, die über dem vorgegebenen Schwellenwert liegen, werden als Signal markiert. Je tiefer der Faktor gewählt wird, desto mehr Signale werden erkannt.

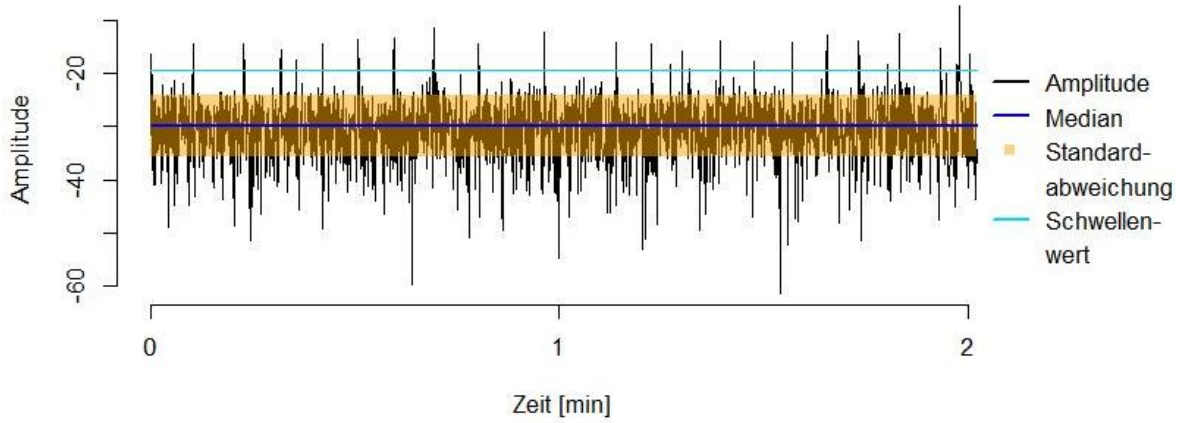


Abbildung 4: Amplitudenwerte eines Frequenzfensters als Zeitsequenz dargestellt. Werte, die den Schwellenwert überschreiten, werden als Signal registriert. Die Darstellung entspricht dem Abschnitt 2 des Übungsfiles, der 22 Rufe enthält.

4. Cluster

In einem ersten Schritt werden alle Zellen, die als Signale markiert wurden und sich in vertikaler Richtung berühren, zu Linien zusammengefügt. Anschliessend werden Linien, die sich horizontal überlagern, zu einer Box zusammengefügt.

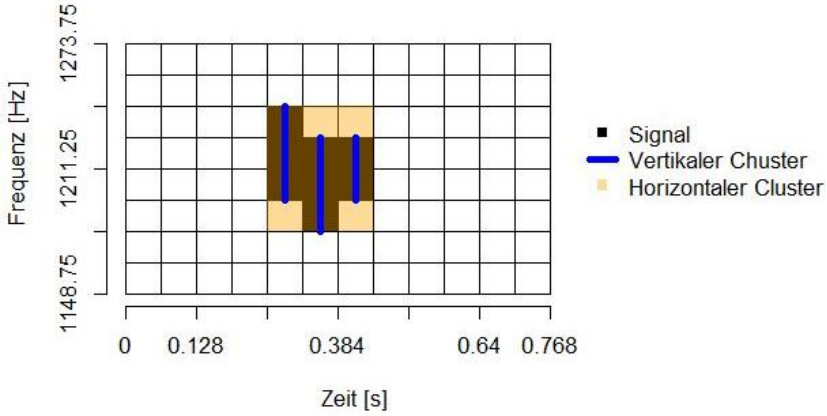


Abbildung 5: Vertikale und horizontaler Cluster an einem einzelnen Ruf. Als Signal markiert sind Zellen mit einem Wert über dem Schwellenwert.

5. Filter

Alle Signale, die folgenden Kriterien entsprechen, werden herausgefiltert:

1. Das Signal dauert länger als 3 Zeitfenster *(max.cluster.length).*
2. Das Signal erstreckt sich in der Frequenz über mehr als 6 Frequenzfenster *(max.cluster.height).*
3. Das Signal erstreckt sich über nur ein Frequenzfenster *(min.cluster.height).*

6. Zusammenfügen von Rufreihen pro Frequenzband

Um zu erkennen, ob eine Reihe von Signalen regelmässig oder zufällig aufgebaut ist, wird die Zeitdifferenz zwischen den einzelnen Signalen ermittelt. Signale, die eine Zeitdifferenz zum nächsten Signal von weniger als 1 s *(threshold.fix.low)* oder mehr als 20 s *(threshold.fix.high)* aufweisen, werden nicht weiter behandelt. Von den verbleibenden Signalen wird der Median der Zeitdifferenz berechnet (innerhalb des 2 min Abschnitts). Signale, die innerhalb eines Auswahlbereichs ober/ unterhalb des Medians liegen, werden zu einer Rufreihe zusammengefügt. Der Grenzwert des Auswahlbereichs wird nach unten durch 60 %

(median.threshold.low) und nach oben durch 160 % *(median.threshold.high)* des Medians definiert. Der Vorgang wird für jedes Frequenzband wiederholt.

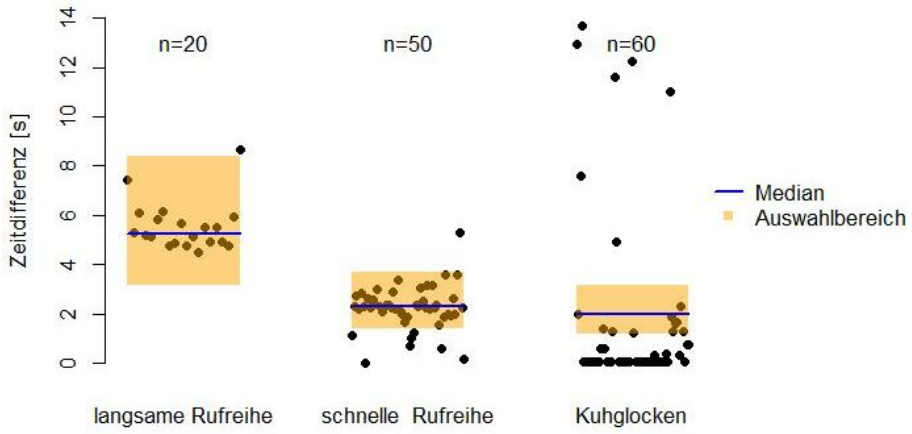


Abbildung 6: Zeitdifferenz von drei Signalreihen. Ein grosser Teil der Signale fällt bei den Rufreihen in den Auswahlbereich, jedoch nur ein kleiner Teil bei den Kuhglocken. Die Signalreihen entsprechen den Abschnitten 1 bis 3 des Übungsfiles.

7. Zusammenfügen der Reihen zu Boxen

Rufsequenzen der verschiedenen Frequenzbänder, die mehr als 5 Rufe *(min.seq.length)* enthalten und sich überlagern, werden zu Boxen zusammengefügt.

8. Generieren des Outputs

Folgende Outputs werden wahlweise generiert und als Raven-Selection-Table abgespeichert:

1. Einzelsignale (nicht sortiert, ob innerhalb einer Rufsequenz)
2. Rufreihe pro Frequenzband
3. Rufsequenz als Box

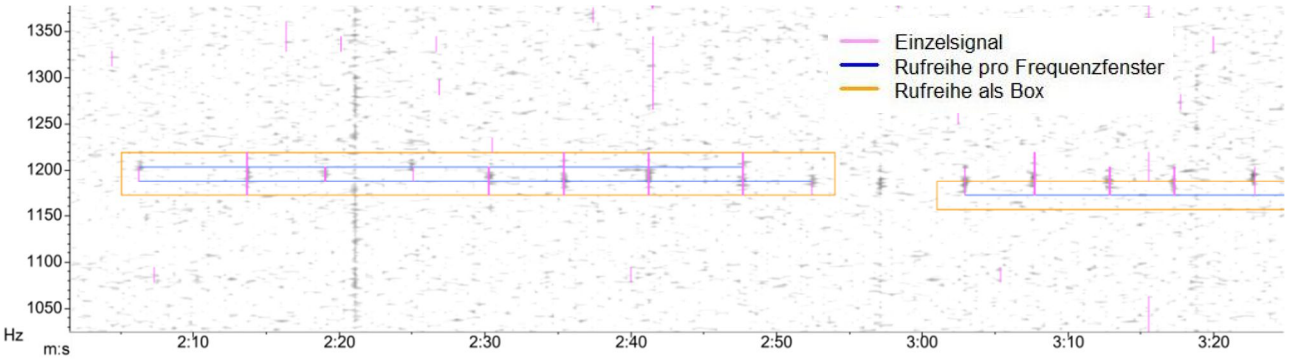


Abbildung 7: Verschiedene Outputs, die die Funktion „find.alob" generiert.

Tabelle 11: Parameter der Funktion „find.alob“.

Beschreibung	Parameter	Voreinstellung	Einheit
Input Audiokanal	*channel*	1	
Abschnitt des Audiofiles, der pro Abschnitt bearbeitet wird	*section.length.min*	2	min
Faktor zur Bestimmung des Schwellenwerts	*threshold.factor*	0.8	Faktor
Maximale Länge der Cluster	*max.cluster.length*	3	Zeitfenster
Maximale Höhe der Cluster	*max.cluster.height*	6	Frequenzfenster
Minimale Höhe der Cluster	*min.cluster.height*	2	Frequenzfenster
Minimaler Zeitunterschied zwischen zwei Rufen	*threshold.fix.low*	1	s
Maximaler Zeitunterschied zwischen zwei Rufen	*threshold.fix.high*	20	s
Faktor zur Bestimmung der maximalen Abweichung der Zeitdifferenz vom Median nach oben	*median.threshold.low*	0.6	Faktor
Faktor zur Bestimmung der maximalen Abweichung der Zeitdifferenz vom Median nach unten	*median.threshold.high*	1.6	Faktor
Mindestlänge einer Rufreihe	*min.seq.length*	5	Rufe
Folgende Outputs werden generiert, wenn die Parametereinstellung „TRUE“ lautet			
Einzelsignal (ungefiltert)	*calls.detected*	FALSE	
Rufreihe pro Frequenzfenster	*lines.call.seq*	FALSE	
Rufreihe als Box	*call.seq*	TRUE	

Da die Filtermethode mit einem Faktor zur Bestimmung des Schwellenwerts *(threshold.factor)* von 0.9 bei den Aufnahmen aus der Mehlbachgrube weniger gut abschnitt als erwartet, habe ich die Aufnahmen zusätzlich mit einem Faktor von 0.8 ausgewertet.

Mit RavenLite habe ich die Ergebnisse der Filtermethode anschliessend kontrolliert. Dazu werden das Audiofile und die entsprechende Raven-Selection-Table aus dem Output in das Programm geladen und mit den Einstellungen in Tabelle 12 visualisiert. Sequenzen ohne Rufe werden entfernt und bei Rufsequenzen wird kontrolliert, ob die gesamte Sequenz erkannt wurde. Unvollständig erkannte Rufreihen lassen sich einfach anpassen. Der Zeitaufwand zum Kontrollieren und Korrigieren der Ergebnisse erfasste ich und verglich ihn mit dem Zeitaufwand für die manuelle Auswertung.

Tabelle 12: Einstellungen zum Paging und zur Visualisierung von Geburtshelferkröten-Rufen in RavenLite.

Paging	
Page size	180 s
Page increment	100 %
Step increment	10 %
Visualisierung	
Views	nur Spektrogramm 1
Set brightness	72
Set contrast	74
Set focus (spectrogram window size)	1300
Lines	1
Zoom X-Achse	Zoom to all X
Zoom Y-Achse	1000-1500 Hz

3.4 Lokalisierung

Ziel der Lokalisierung ist es, mittels synchronisierter Aufnahmen den Standort von Rufern zu ermitteln und darauf auf die Anzahl Rufer in einem Gebiet zu schliessen. Die Lokalisierung von Vokalisationen ist eine bekannte Methode, die sich auch für Geburtshelferkröten einsetzen lassen sollte (Wilson et al., 2013). Um die Genauigkeit der Methode einzuschätzen, führte ich zuerst einen Laborversuch mit abgespielten Rufen durch. Anschliessend wurde die Methode im Feld erprobt. Als Studienobjekt dienten die Geburtshelferkröten-Bestände in der Region Glaubenbielen, Kanton Obwalden.

Um Rufe in einer Tonaufnahme zu lokalisieren, müssen die verwendeten Aufnahmegeräte äusserst exakt synchronisiert sein. Dies ist entweder möglich, in dem Mikrofone mit einem zentralen Aufnahmegerät durch Kabel verbunden sind oder indem die Uhrzeit von mehreren Aufnahmegeräten über einen GPS-Empfänger synchronisiert wird. Hier wird die letztere Methode angewandt. Weiter müssen die Standorte der Aufnahmegeräte im Raum exakt bekannt sein. (Wilson et al., 2013)

3.4.1 Material

Tabelle 13: Verwendete Messgeräte zur Lokalisierung von Schallquellen.

Messgerät	Anzahl
Songmeter SM3 Aufnahmegerät	6
GPS-Empfänger zu SM3	6
Oregon 650t GPS-Navigationsgerät	1
Bstgo 100M Laser-Entfernungsmesser	1
testo 816 Schallpegelmesser	1

3.4.2 Laborversuch

In einem ersten Schritt sollte sich unter „Laborbedingungen“ zeigen, ob sich die verwendeten Aufnahmegeräte zur Lokalisierung von abgespielten Geburtshelferkrötenrufe eignen. Weiter diente dieser Schritt dazu, ein R-Skript zu entwickeln, das die Eintreff-Zeitdifferenzen von Rufen automatisch berechnete und die Lokalisierung mit Hilfe des Packages Sound Finder durchführte.

Dazu habe ich ein Versuchsaufbau mit 3 Aufnahmegeräten erstellt. Die Geräte waren in einem Dreieck mit Seitenlängen von $\overline{AB}$ = 34.8 m, $\overline{BC}$ = 32.5 m und $\overline{CA}$ = 32.2 m auf 50 cm hohen Holzpfosten montiert. Da die Koordinaten der GPS-Empfänger zur Synchronisation nicht genügend genaue Daten lieferten, habe ich die Standorte der Aufnahmegeräte mit einem Distanzmessgerät vermessen. Dazu habe ich zuerst von zwei Fixpunkten im Gelände, die auf dem Luftbild von Swisstopo gut zu erkennen sind, die Koordinaten herausgelesen. Mit der Distanz von diesen Punkten zu jedem Standort, liessen sich anschliessend mit Hilfe des R-Skrips im Anhang XI die Koordinaten der Standorte berechnen und als .csv-Datei abspeichern. Bei einem ebenfalls vermessenen Punkt habe ich 25 Rufe einer Geburtshelferkröte abgespielt. Die Lautstärke des Lautsprechers habe ich mithilfe des Schallpegelmessers auf maximal 84.6 dB in 15 cm Abstand zum Lautsprecher eingestellt (Marti, 2019). Um sicherzustellen, dass die Synchronisation der Aufnahmegeräte via GPS-Empfänger ordnungsgemäss funktioniert, ist eine Vorlaufzeit von einigen Minuten nötig. Bleibt das Gerät allerdings länger als eine Minute im Pause-Modus, schaltet es sich aus. Es hat sich gezeigt, dass ein Vorlauf von einer Minute Pause, einer Minute Aufnahme und einer weiteren Minute Pause vor der Aufnahme optimal ist. Die Einstellungen der Aufnahmegeräte sind in Tabelle 14 aufgeführt.

Tabelle 14: Einstellungen der SM3 Aufnahmegeräte.

Einstellungen	
Samplingrate (sample rate)	24 kHz
Samplingtiefe (sample size)	16 Bit
Aufnahmezeit	1 min (1 min Vorlauf, 1 min Aufnahme, 1 min Pause, 1 min Hauptaufnahme)

Um die Eintreff-Zeitdifferenzen von Geräuschen zu berechnen, wurden zuerst einzelne Rufe mit dem Programm RavenLite gesucht und markiert. Wilson et al. (2013) verwenden anschliessend eine Kreuzkorrelationsmethode des kostenpflichtigen Programms RavenPro, um den zeitlichen Unterschied der Rufe zu berechnen. R bietet mit der Funktion ccf ebenfalls eine Methode der Kreuzkorrelation an. Mit dieser konnte ich aber keine verwendbaren Resultate erzielen. Deshalb entwickelte ich eine eigene Methode. Dabei werden die markierten Rufe in Abschnitten von 0.4 s ausgeschnitten und mit einer Fouriertransformation in eine Tabelle umgewandelt. Die Tabelle wird auf einen dem Ruf entsprechenden Frequenzbereich reduziert und in eine Zeitsequenz umgewandelt. Nun wird gemessen, wann die erste Zeitsequenz einen bestimmten Schwellenwert überschreitet. Bei der Überschreitung wird die Zeit auf null gesetzt, und die Zeitdifferenz bis zur Überschreitung der folgenden Zeitsequenzen gemessen. Diese Zeitdifferenzen entsprechen den Eintreff-Zeitdifferenzen, die anschliessend zur Lokalisierung dienen (R-Skript im Anhang XI).

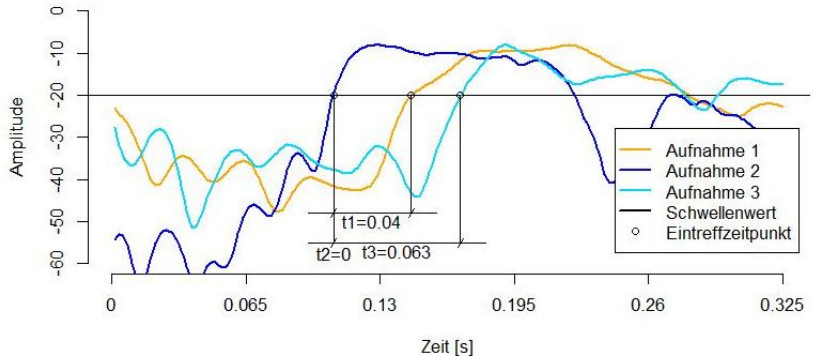


Abbildung 8: Ausmessung der Eintreff-Zeitdifferenzen eines Geburtshelferkrötenrufs in drei synchronisierten Aufnahmen. Abgespeichert werden die Werte t1, t2 und t3.

Nachdem eine Methode gefunden war, die es erlaubt, Rufe zu lokalisieren, habe ich an 4 verschiedenen Punkten in der Versuchsanordnung je 20 Rufe abgespielt, mit dem Ziel, eine Clusteranalyse zu entwickeln, die es erlaubt, einzelne Rufer zu unterscheiden und so auf die Anzahl Rufer im Gebiet zurück zu schliessen (R-Skript im Anhang XI). Die Clusteranalyse basiert auf einer hierarchischen Clusteranalyse der Distanz zwischen den einzelnen berechneten Punkten. Dabei wird ein Cluster erkannt, sobald mindestens 5 Punkte in einem Umkreis von 15 m liegen. Diese Parameter müssen je nach Anzahl und Genauigkeit der Punkte angepasst werden.

3.4.3 Feldversuch

Den Feldversuch führte ich im Gebiet Glaubenbielen im Kanton Obwalden durch. Es handelt sich um ein Amphibienlaichgewässer von nationaler Bedeutung mit einem grossen Bestand an Geburtshelferkröten (Bundesamt für Umwelt BAFU, 2017). Für den Versuch wählte ich ein relativ flaches und gut zugängliches Gebiet nordwestlich des Ribiseelis, wo wiederholt zahlreiche Rufer festgestellt wurden. Die sechs verwendeten Aufnahmegeräte ordnete ich in zwei, dem Gelände angepassten Reihen an. Damit konnte eine Fläche von 915.6 m^2 mit einer durchschnittlichen Kantenlänge von 25.2 m (14.1 bis 24.8 m) abgedeckt werden. Dies entspricht grob einer Vervielfältigung des Dreiecks, das im Laborversuch angewandt wurde. Die Standorte der Aufnahmegeräte vermass ich mit derselben Methode, wie im Laborversuch

beschrieben. Ergänzend nahm ich die Koordinaten mit einem GPS-Handgerät auf (zwei Punkte pro Gerät) und wertete die Daten der GPS-Empfänger der Aufnahmegeräte aus (ca. alle 30 s ein Punkt pro Gerät). Dies, um die verschiedenen Methoden zu vergleichen.

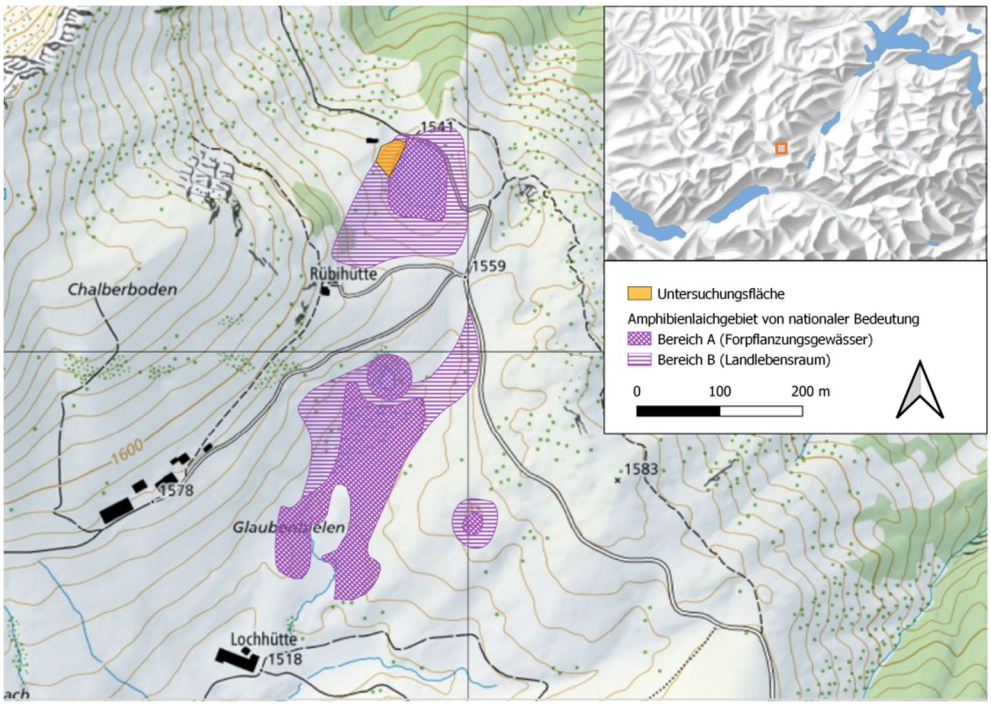


Abbildung 9: Übersicht über das Amphibienlaichgebiet Glaubenbielen mit der Untersuchungsfläche. Quelle: Bundesamt für Landestopographie swisstopo.

Die Aufnahmen fanden während zwei Nächten vom 4. bis am 6. Juli 2020 jeweils von 19:00 bis 06:00 Uhr statt. Die Aufnahmen wurden in jeweils einstündigen Dateien abgespeichert. Die Einstellungen entsprachen den Einstellungen, die ich für den Labortest verwendete. Die Auswertung erfolgte analog dem Laborversuch anhand von einminütigen Ausschnitten.

4 Resultate

4.1 Automatische Auswertung

4.1.1 Ishmael

Das Programm Ishmael fand mit der Methode „Energy sum detection“ 199 Signale, verteilt auf alle vier Abschnitte, wovon die meisten Entdeckungen im Abschnitt 4 mit 100 Signalen liegen. Die Lücken in Abschnitt 4 ergaben sich praktisch ausschliesslich aus dem definierten Mindestabstand zwischen zwei Signalen von einer Sekunde (Abbildung 10).

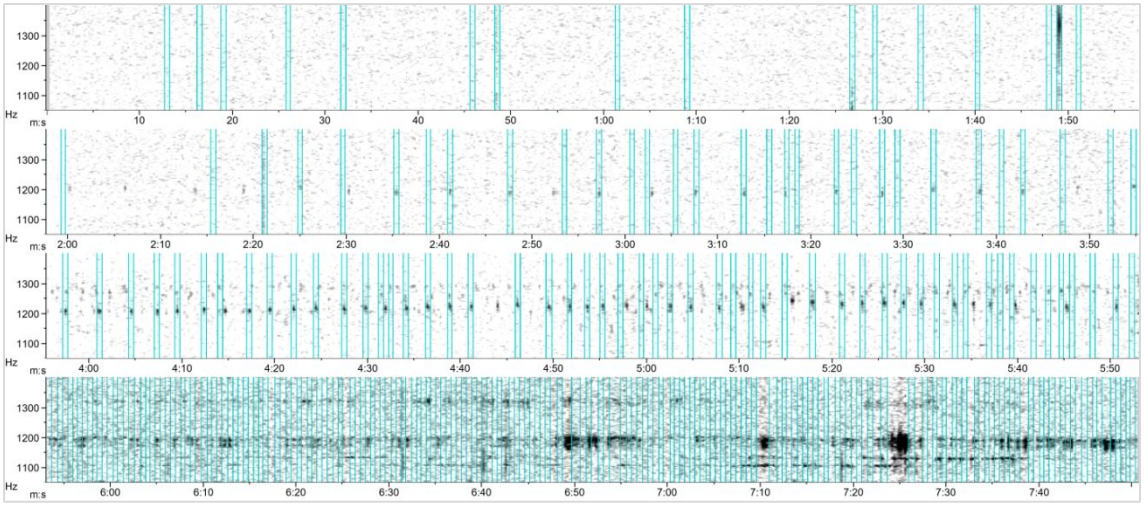


Abbildung 10: Automatisch generierte Markierungen mit dem Programm Ishmael, Methode „Energy sum detection". Abbildung erstellt mit RavenLite.

Mit der Methode „Spectrogram correlation" fand das Programm 49 Signale. Praktisch alle Signale entsprechen einem Ruf (Abbildung 11). Die Frequenzkontur ist nicht universell anwendbar. Bereits bei einer Erhöhung der Start- und Endfrequenz um 50 Hz, werden keine Rufe mehr gefunden.

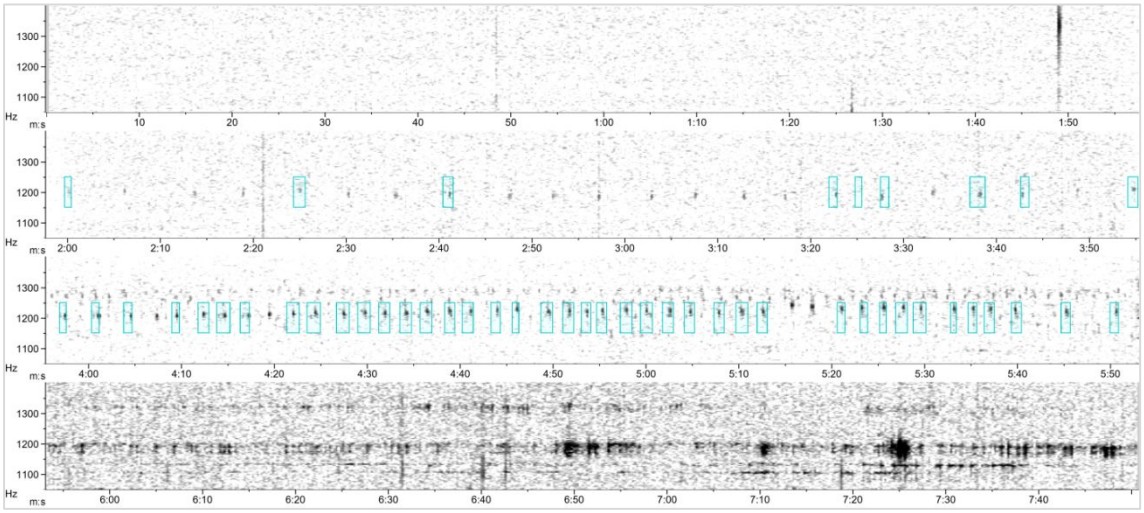


Abbildung 11: Automatisch generierte Markierungen mit dem Programm Ishmael, Methode „Spectrogram correlation". Abbildung erstellt mit RavenLite.

Die Kreuzkorrelation mit dem Kernel („matched filter") brachte insgesamt 167 erkannte Signale. Ein Grossteil davon (104) liegt in Abschnitt 4 (Abbildung 12).

Alle drei Methoden innerhalb des Programms Ishmael erfüllen die eingangs gestellten Kriterien nicht (Tabelle 4). Deshalb wurden mit Ishmael keine Langzeitaufnahmen analysiert.

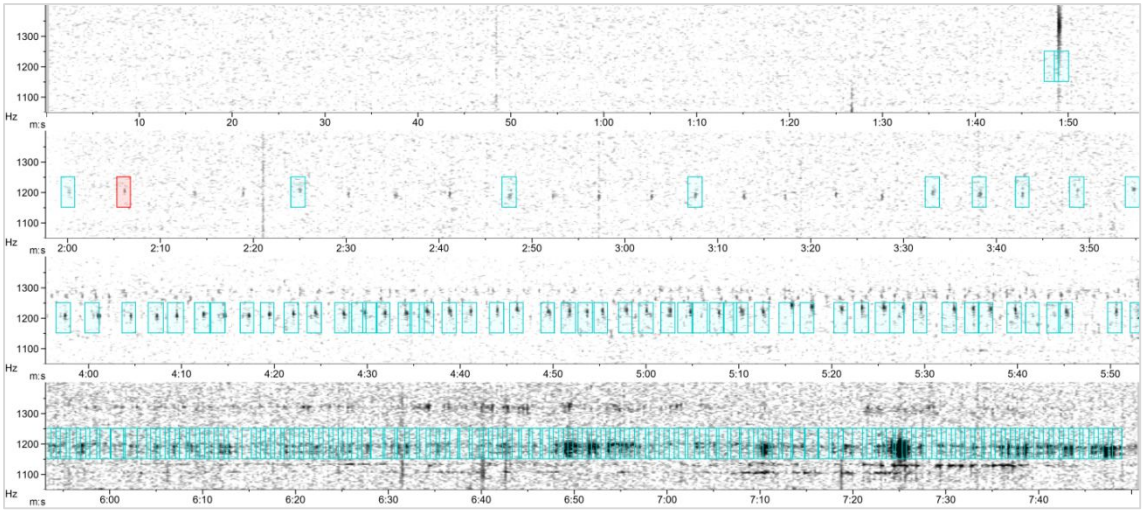


Abbildung 12: Automatisch generierte Markierungen mit dem Programm Ishmael, Methode „matched filter“. Rot markiert ist der als Kernel verwendete Ausschnitt. Abbildung erstellt mit RavenLite.

4.1.2 Kaleidoscope

Das Programm fand 51 Signale, davon liegen 44 in Abschnitt 3 und 7 in Abschnitt 4. Die etwas leiseren Rufe in Abschnitt 2 wurden nicht erkannt (Abbildung 13). Bei diesem Auswertungsschritt hat das Programm noch keine Einteilung anhand des Klassifikators in TP und FP vorgenommen.

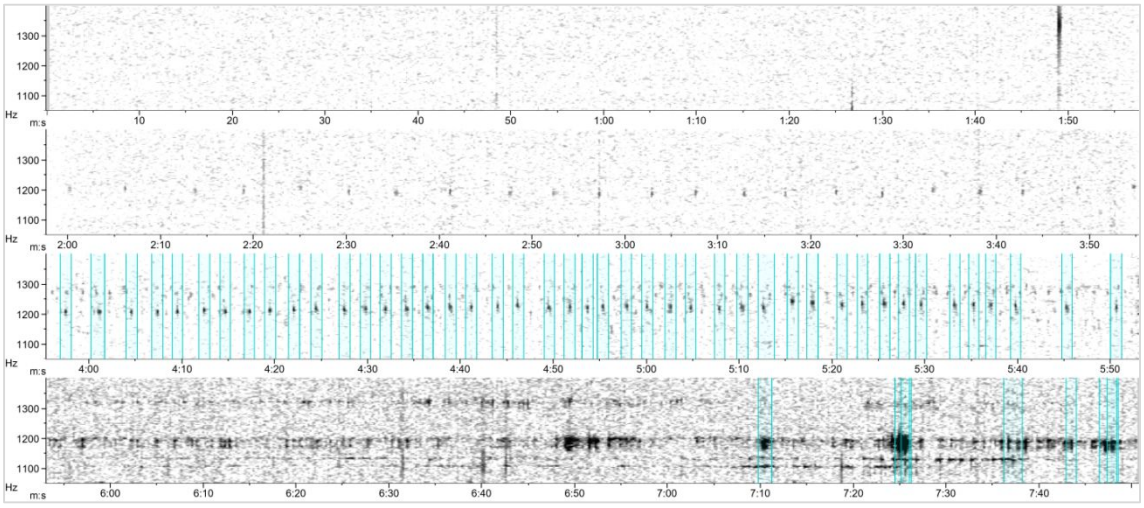


Abbildung 13: Automatisch generierte Markierungen mit dem Programm Kaleidoscope. Eine Einteilung in verschiedene Cluster ist hier noch nicht erfolgt. Abbildung erstellt mit RavenLite.

Beim Test des erstellten Klassifikators fand das Programm 1272 Signale, davon wurden 1176 als Rufe eingestuft. Davon handelt es sich bei 1137 (96.7 %) um richtig Positive und bei 39 (3.3 %) um falsch Positive.

Die Auswertung der Aufnahmen des Geräts OW03 ergab 1069 Signale. Davon hat das Programm 991 als Rufe eingeteilt, bei 27 handelt es sich tatsächlich um Rufe (TP). Die falsch Negativen lassen sich mit dem Programm nicht direkt analysieren.

In den Aufnahmen aus der Dickbangrube fand Kaleidoscope Rufsequenzen von 38.6 h Dauer. Davon sind 36.7 h tatsächliche Rufreihen (TP) und 4.6 h wurden nicht erkannt (FN) (Tabelle 14, Abbildung 14).

Tabelle 14: Wahrheitsmatrix der Aufnahmen aus der Dickbangrube, ausgewertet mit Kaleidoskop.

	manuelle Auswertung	
automatische Auswertung	positiv	negativ
positiv	2 203.3 min	112.6 min
negativ	276.9 min	1 547.2 min
Sensitivität/Spezifität	88.8 %	93.2 %

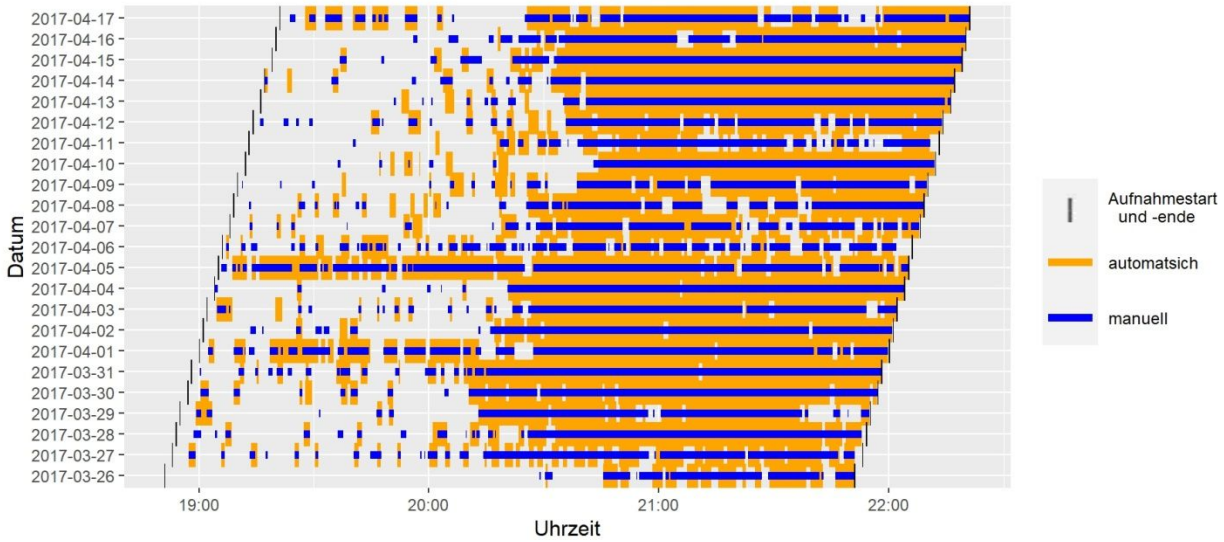


Abbildung 14: Vergleich der manuell und automatisch mit dem Programm Kaleidoscope ausgewerteten Rufsequenzen der Aufnahmen aus der Dickbangrube. Interpretation: orange und blau = richtig Positive (TP), nur orange = falsch Positive (FP), nur blau = falsch Negative (FN), keine Markierung = richtig Negative (TN).

4.1.3 Filtermethode

Die Filtermethode erkennt nicht einzelne Rufe, sondern ganze Rufreihen. Im Übungsfile fand der Detektor 4 Rufreihen in den Abschnitten 2 und 3. Diese enthalten einen Grossteil aller Rufe der Audiodatei (Abbildung 15).

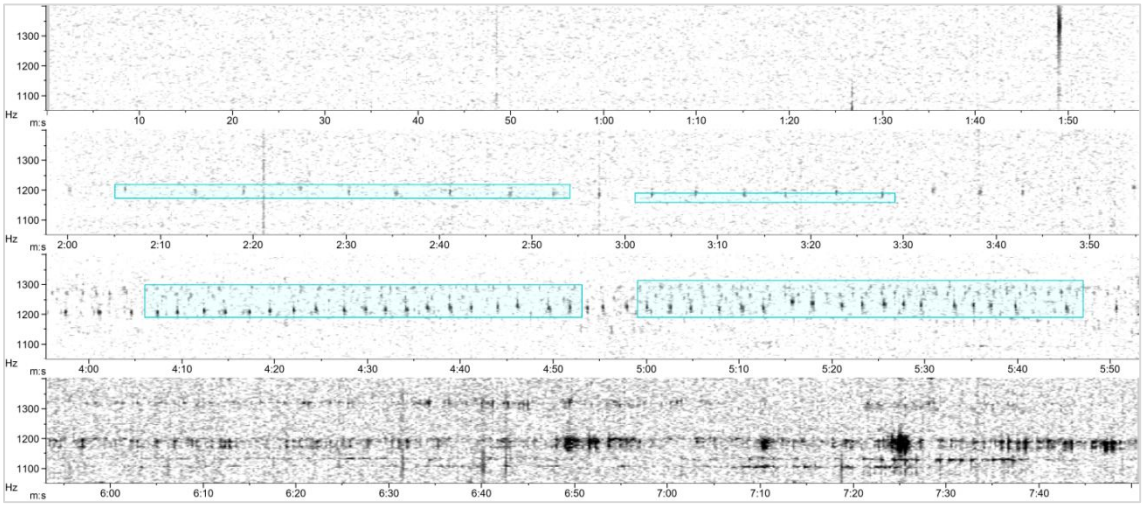


Abbildung 15: Automatisch generierte Markierungen mit der Filtermethode (Schwellenwert 0.9). Abbildung erstellt mit RavenLite.

Mit einem Schwellenwert von 0.9 fand die Filtermethode in den Aufnahmen des Geräts OW03 Rufsequenzen mit einer Dauer 2.1 h, wovon 1.2 h tatsächliche Rufreihen sind. Weitere 9.9 h mit Rufreihen fand

der Detektor nicht (Tabelle 15). Bei der anschliessenden Bearbeitung erhöhte sich die Dauer der gefundenen Rufreihen um 2.0 h auf 3.2 h. Falsch Positive konnten praktisch ausgeschlossen werden und die nicht gefundenen Rufsequenzen (FN) reduzierten sich auf 8.0 h (Tabelle 16, Abbildung 16). In den Aufnahmen des Geräts OW04 fand der Detektor 0.7 h falsch positive Rufsequenzen. Die Bearbeitung der Ergebnisse beider Geräte dauerte 93 Minuten.

Tabelle 15: Wahrheitsmatrix der Aufnahmen OW03, ausgewertet mit der Filtermethode 0.9 (unbearbeitet).

automatische Auswertung	manuelle Auswertung positiv	negativ
positiv	72.0 min	52.0 min
negativ	595.9 min	31 080.1 min
Sensitivität/Spezifität	10.8 %	99.8 %

Tabelle 16: Wahrheitsmatrix der Aufnahmen OW03, ausgewertet mit der Filtermethode 0.9 (bearbeitet).

automatische Auswertung	manuelle Auswertung positiv	negativ
positiv	186.4 min	2.7 min
negativ	481.5 min	31129.5 min
Sensitivität/Spezifität	27.9 %	100 %

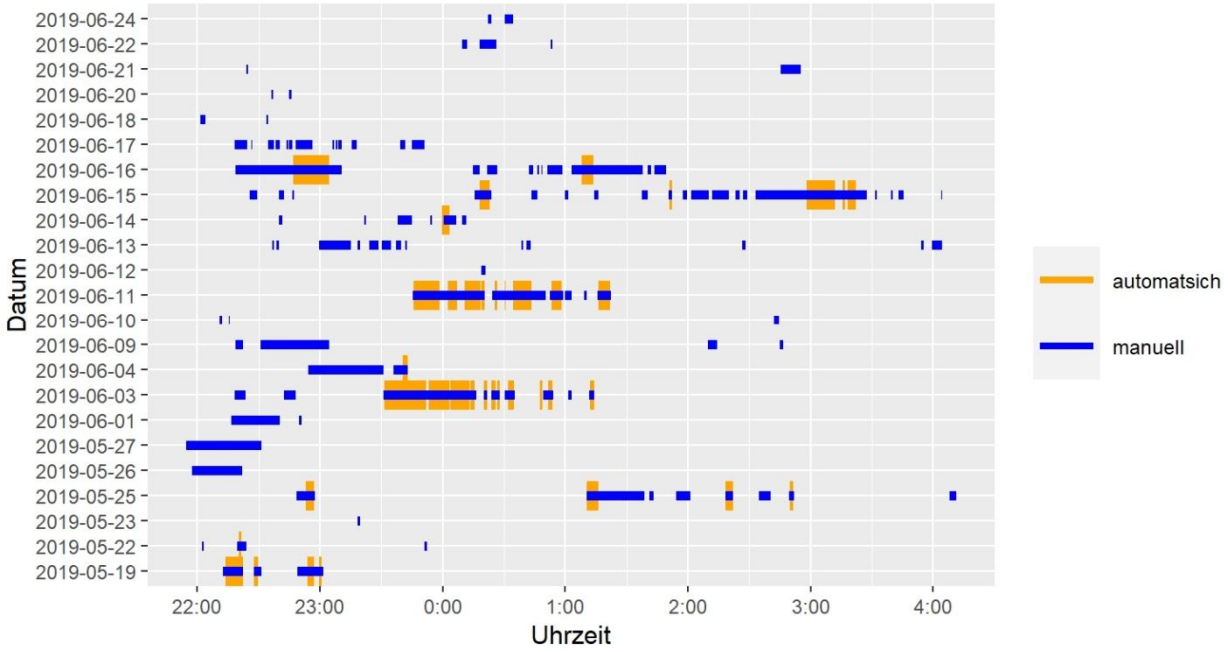


Abbildung 16: Vergleich der manuell und automatisch mit der Filtermethode (Schwellenwert 0.9, bearbeitet) ausgewerteten Rufsequenzen der Aufnahmen aus der Mehlbachgrube (Gerät OW03).

Mit einem Schwellenwert von 0.8 fand die Filtermethode in den Aufnahmen des Geräts OW03 Rufsequenzen mit einer Dauer 11.1 h, wovon 2.1 h tatsächliche Rufreihen waren. Weitere 9.0 h mit Rufreihen fand der Detektor nicht (Tabelle 17). Bei der anschliessenden Bearbeitung erhöhte sich die Dauer der gefundenen Rufreihen um 6.1 h auf 8.2 h. Falsch Positive reduzierten sich auf 1.1 h und die nicht gefundenen Rufsequenzen (FN) reduzierten sich auf 3.0 h (Tabelle 18, Abbildung 17). In den Aufnahmen des Geräts OW04 fand der Detektor 17.1 h falsch positive Rufsequenzen. Die Bearbeitung der Ergebnisse beider Geräte dauerte 131 Minuten.

Tabelle 17: Wahrheitsmatrix der Aufnahmen OW03, ausgewertet mit der Filtermethode 0.9 (unbearbeitet).

automatische Auswertung	manuelle Auswertung positiv	negativ
positiv	126.7 min	543.9 min
negativ	541.2 min	30’588.3 min
Sensitivität/Spezifität	19.0 %	98.3 %

Tabelle 18: Wahrheitsmatrix der Aufnahmen OW03, ausgewertet mit der Filtermethode 0.9 (bearbeitet).

automatische Auswertung	manuelle Auswertung positiv	negativ
positiv	490.4 min	66.9 min
negativ	177.5 min	31’065.3 min
Sensitivität/Spezifität	73.4 %	99.8 %

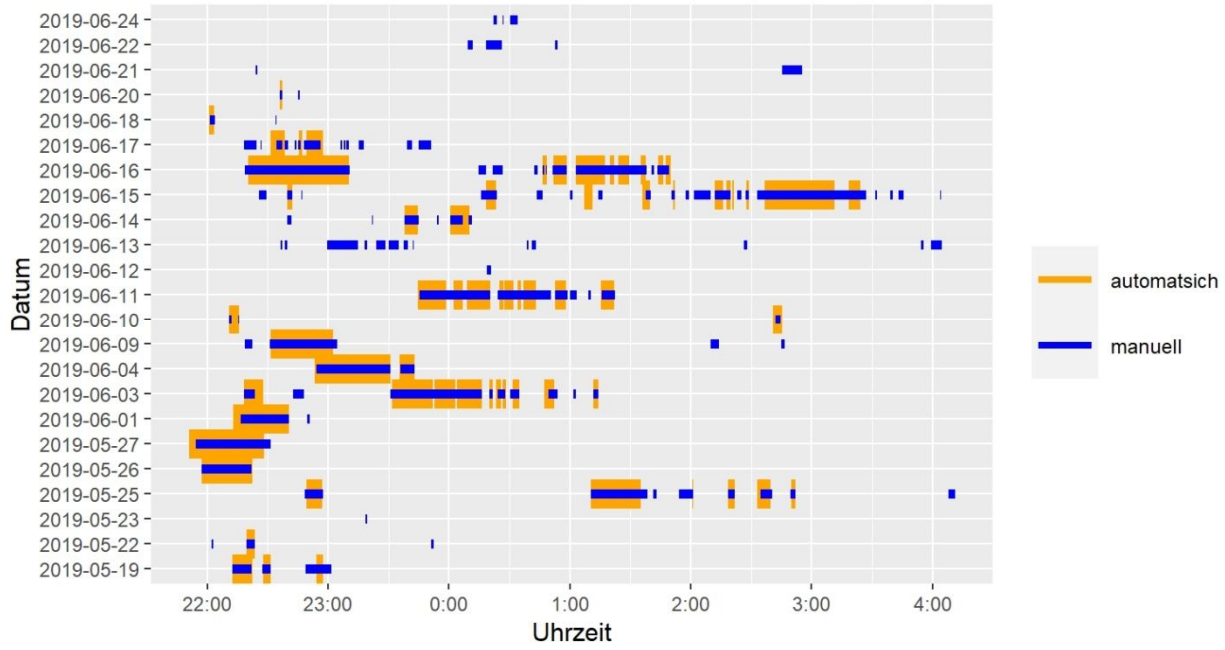


Abbildung 17: Vergleich der manuell und automatisch mit der Filtermethode (Schwellenwert 0.8, bearbeitet) ausgewerteten Rufsequenzen der Aufnahmen aus der Mehlbachgrube (Gerät OW03).

In den Aufnahmen aus der Dickbangrube fand die Filtermethode ohne manuelle Nachkontrolle Rufsequenzen mit 25.3 h Dauer, davon waren 22.7 h tatsächliche Rufreihen (TP) und 18.6 h wurden nicht erkannt (FN) (Tabelle 19, Abbildung 18).

Tabelle 19: Wahrheitsmatrix der Aufnahmen aus der Dickbangrube, ausgewertet mit der Filtermethode 0.9 (unbearbeitet).

	manuelle Auswertung	
automatische Auswertung	positiv	negativ
positiv	1'361.0 min	158.3 min
negativ	1’119.2 min	1’501.5 min
Sensitivität/Spezifität	54.9 %	90.5 %

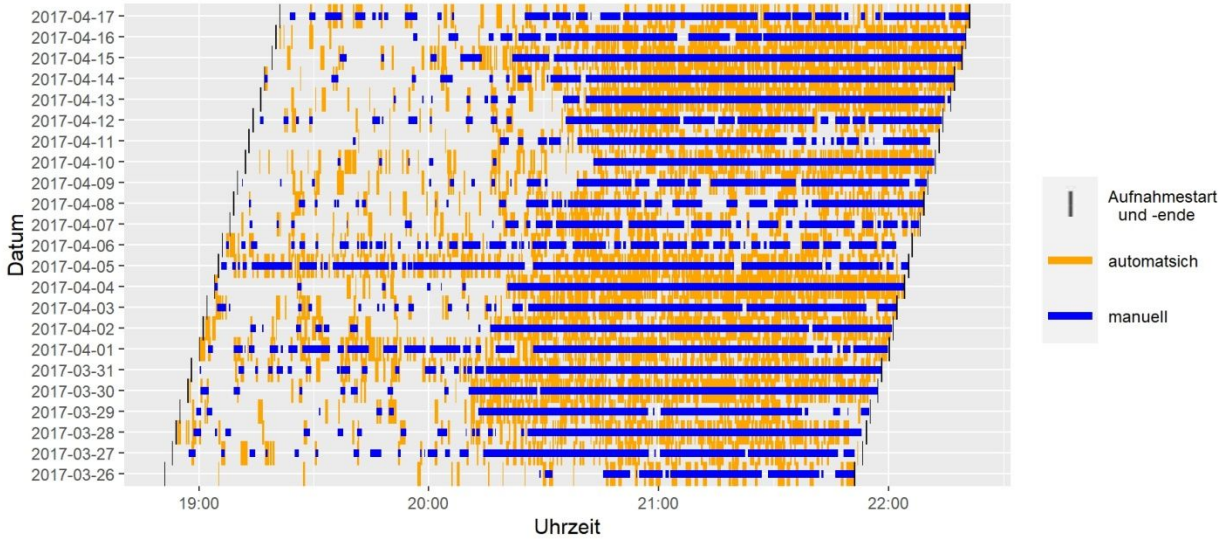


Abbildung 18: Vergleich der manuell und automatisch mit der Filtermethode (Schwellenwert 0.9, unbearbeitet) ausgewerteten Rufsequenzen der Aufnahmen aus der Dickbangrube.

4.1.4 Vergleich der Methoden

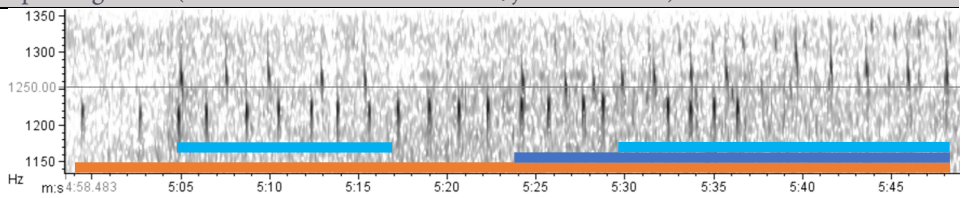
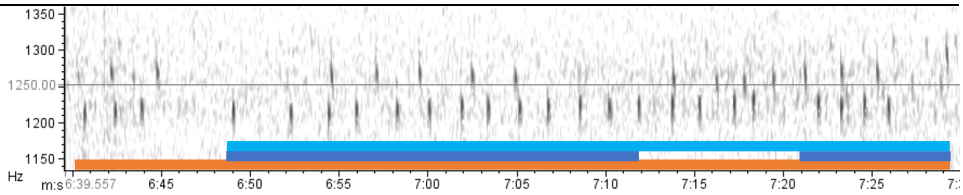
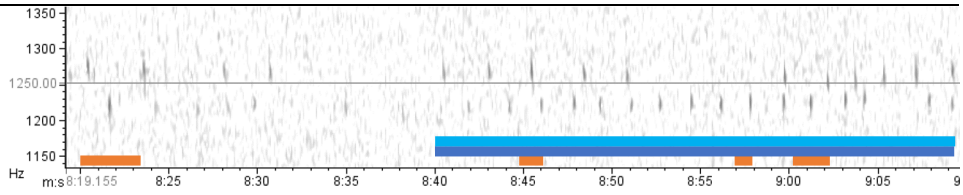
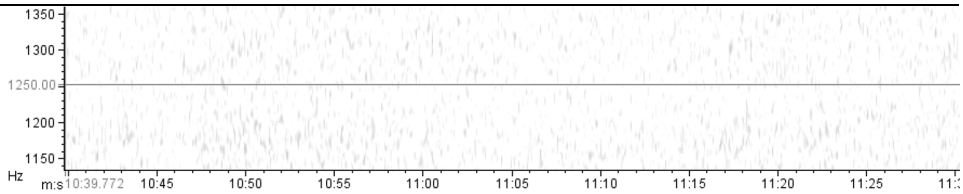
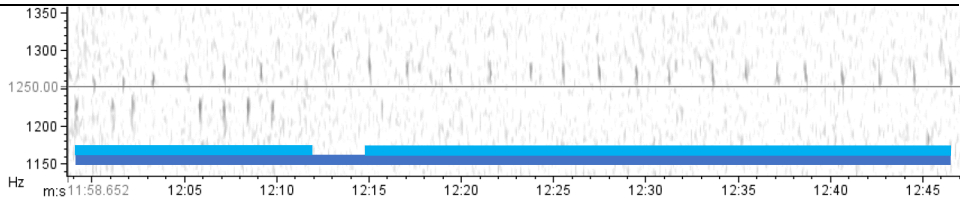
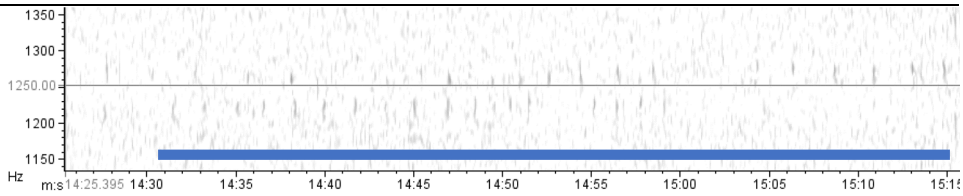
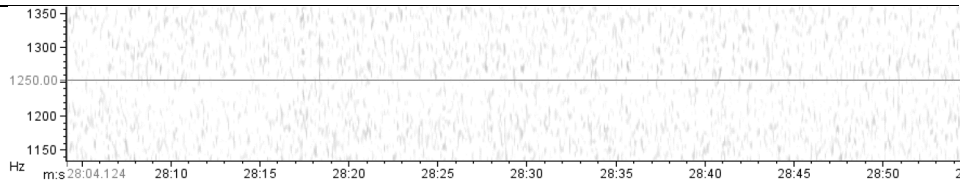
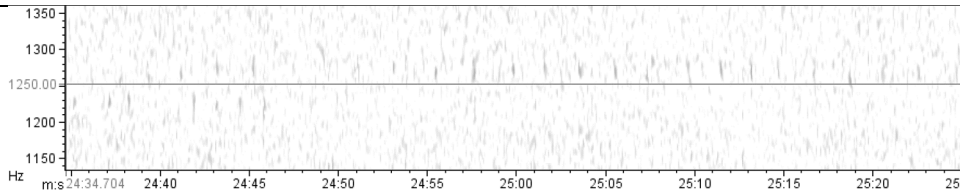
In der Vergleichstabelle (Tabelle 20) sind die tatsächlichen Gegebenheiten (Ground truth) des Übungsfiles sowie die Ergebnisse der verschiedenen Methoden dargestellt.

Tabelle 20: Vergleich der verschiedenen Detektoren mit den tatsächlichen Gegebenheiten (Ground truth).

	Ground truth	Ishmael Energy sum	Ishmael Spectrogram correlation	Ishmael Matched filter	Kaleidoscope	Filtermethode (Schwellenwert 0.8)
Abschnitt 1	0	16	0	2	0	0
Abschnitt 2 TP	22	14	8	10	0	15
Abschnitt 2 FP	0	16	1	0	0	0
Abschnitt 2 FN	0	8	14	12	22	7
Abschnitt 3	durchgehend Rufe	durchgehend Rufe (53) «gut»	durchgehend Rufe (40) «gut»	durchgehend Rufe (51) «gut»	durchgehend Rufe (44) «sehr gut»	durchgehende Rufreihe «sehr gut»
Abschnitt 4	0	99	0	104	7	0
Rechendauer	-	4 s	4 s	5 s	1 s	20 s
Info zur Ruffrequenz	-	nein	ja	nein	nein	ja
Info zur Rufdauer	-	ja	ja	nein	ja	ja
Kriterien erfüllt	-	nein	nein	nein	ja	ja
Nicht erfüllte Kriterien (Tabelle 4)	-	1, 3	(2), 4	(2), 3	(2)	-
Kommentar	-	-	Parameter nicht universell, vor allem deutliche Rufe gefunden	vor allem deutliche Rufe gefunden	nur deutliche Rufe gefunden	-

Der Distanztest zeigt, wie die Detektoren Rufreihen, die in verschiedenen Distanzen zum Aufnahmegerät abgespielt wurden, erkennen. Kaleidoscope fand Rufe bis maximal 20 m Entfernung. Bei der Filtermethode hängt die noch bearbeitbare Reichweite vom Schwellenwert (threshold.factor) ab. Mit einem Schwellenwert von 0.8 können Rufe bis zu einer Distanz von 60 m Entfernung erkannt werden. Topographische Hindernisse können die Reichweite stark einschränken (Tabelle 21).

Tabelle 21: Darstellung der Resultate des Distanztests: orange = erkannte Rufe mit Kaleidoscope, dunkelblau = erkannte Rufe mit Filtermethode (Schwellenwert 0.8), hellblau = erkannte Rufe mit Filtermethode (Schwellenwert 0.9).

Distanz	Sensitivität	Spektrogramm (Bildschirmfoto aus RavenLite, jeweils ca. 50 s)
5 m	Filtermethode 0.9: 62 % Filtermethode 0.8: 50 % Kaleidoscope: 100 %	
10 m	Filtermethode 0.9: 65 % Filtermethode 0.8: 82 % Kaleidoscope: 100 %	
20 m	Filtermethode 0.9: 69 % Filtermethode 0.8: 69 % Kaleidoscope: 11 %	
50 m Abspielgerät am Boden	Filtermethode: 0 % Kaleidoscope: 0 % Nicht hörbar →Topografie!	
50 m Abspielgerät 1 m über Boden	Filtermethode 0.9: 93 % Filtermethode 0.8: 100 % Kaleidoscope: 0 %	
60 m Abspielgerät 1 m über Boden	Filtermethode 0.9: 0 % Filtermethode 0.8: 91 % Kaleidoscope: 0 %	
80 m	Filtermethode 0.9: 0 % Filtermethode 0.8: 0 % Kaleidoscope: 0 %	
130 m	Filtermethode 0.9: 0 % Filtermethode 0.8: 0 % Kaleidoscope: 0 %	

Der Aufwand zur manuellen Nachbearbeitung der Resultate der Filtermethode betrug 10.8 respektive 15.2 % im Vergleich zur manuellen Auswertung mit RavenLite (Tabelle 22). Dabei wurden 23.9 respektive 73.4 % der Rufreihen gefunden (Tabelle 23).

Tabelle 22: Vergleich der Rechendauer und dem Aufwand zur manuellen Bearbeitung bei der Filtermethode (Schwellenwert 0.8 und 0.9) und der manuellen Auswertung.

Aufnahmegerät	Methode	Rechendauer [min]	Bearbeitung [min]	Vergleich zur manuellen Auswertung
OW03+OW04	manuell		864	100 %
OW03+OW04	Filtermethode 0.9	287.4	93	10.8 %
OW03+OW04	Filtermethode 0.8	480.6	131	15.2 %

Ein Vergleich der wichtigsten Resultate ist in Tabelle 23 dargestellt.

Tabelle 23: Zusammenstellung der Resultate der verschiedenen Aufnahmen und Auswertungsmethoden.

Aufnahme	Kaleidoscope		Filtermethode 0.9		Filtermethode 0.8	
Mehlbachgrube (OW03)	Rufe (TP)	27	Sensitivität	23.9 %	Sensitivität	73.4 %
			Spezifität	100 %	Spezifität	99.8 %
			Aufwand	10.8 %	Aufwand	15.2 %
Dickbangrube	Sensitivität	88.8 %	Sensitivität	54.9 %		
	Spezifität	93.2 %	Spezifität	90.5 %		
Distanztest	Maximale Distanz	20 m	Maximale Distanz	50 m	Maximale Distanz	60 m

4.2 Lokalisierung

4.2.1 Laborversuch

Im ersten Versuch mit 25 abgespielten Rufen von einer Schallquelle, konnte das Programm Sound Finder 15 Rufe mit einem Fehlerwert unter 1 m lokalisieren. Der Mittelpunkt (Median in x- und y-Richtung) der Punkte liegt 2.4 m von der Schallquelle entfernt. Die Standardabweichung beträgt in x-Richtung 3.6 m und in y-Richtung 4.8 m.

Abbildung 19: Resultat der Lokalisierung von 25 abgespielten Rufen. Quelle: Bundesamt für Landestopographie swisstopo, Abbildung erstellt mit R.

Im zweiten Versuch mit je 20 abgespielten Rufen von vier Standorten aus, identifizierte das Programm 4 Cluster. Die Entfernung der Mittelpunkte zu den Schallquellen und die Standardabweichung sind in Tabelle 24 aufgeführt.

Abbildung 20: Resultate der Lokalisierung von je 20 an vier Standorte abgespielten Rufen. Quelle: Bundesamt für Landestopographie swisstopo, Abbildung erstellt mit R.

Tabelle 24: Abweichung von der Schallquelle und Streuung der Cluster im Laborversuch.

Cluster	Distanz zur Schallquelle	Standardabweichung in x-Richtung	Standardabweichung in y-Richtung
1	3.5 m	1.8 m	3.9 m
2	4.4 m	2.5 m	3.8 m
3	4.2 m	2.0 m	1.6 m
4	2.1 m	2.5 m	1.5 m

4.2.2 Feldversuch

Tabelle 25 zeigt die Abweichung der drei Vermessungsmethoden. Die Werte stellen die Abweichungen zu den im Luftbild gesetzten Punkten dar. Gerät 3 und 6 befinden sich direkt bei den Fixpunkten. Beim GPS-Handgerät entspricht die Abweichung der durchschnittlichen Abweichung der beiden aufgenommenen Punkte pro Gerät. Bei den GPS-Empfänger der Aufnahmegeräte entspricht die Abweichung dem median der Abweichung aller 599 aufgenommen Punkte pro Gerät.

Tabelle 25: Vergleich der Verschiedenen Methoden zu Vermessung der Standorte der Aufnahmegeräte

Gerät	Distanz Messung	GPS-Handgerät	GPS-Empfänger Aufnahmegerät
1	0.5 m	7.2 m	5.3 m
2	0.2 m	4.0 m	0.5 m
3	-	5.7 m	1.4 m
4	0.3 m	3.6 m	7.6 m
5	0.4 m	3.5 m	5.2 m
6	-	3.6 m	0.9 m

Die Aufnahmen der sechs Geräte unterschieden sich zum Teil stark. Eine Abbildung eines 30 s Ausschnitts aller Geräte ist im Anhang XII zu finden. Im ausgewerteten 1 min Ausschnitt fanden sich 182 Rufe (3 Rufe pro Sekunde). Davon waren 65 Rufe auf drei oder mehr Geräten erkennbar und konnten zur

Lokalisierung verwendet werden. Die Lokalisierung der Rufe wiesen hohe Fehlerwerte auf. In Abbildung 21 sind diese als Boxplot dargestellt.

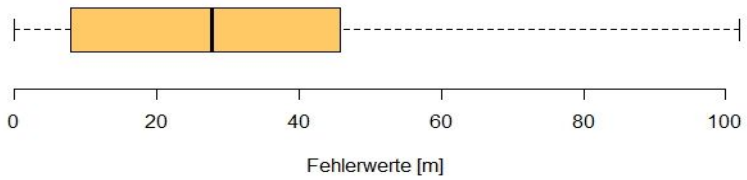


Abbildung 21: Boxplot der Fehlerwerte von 182 lokalisierten Rufen.

Die berechneten Schallquellen bilteten keine Cluster und waren stark verstreut. Einzig einige Punkte mit relativ tiefem Fehlerwert konzentrierten sich rund um das Aufnahmegerät 2.

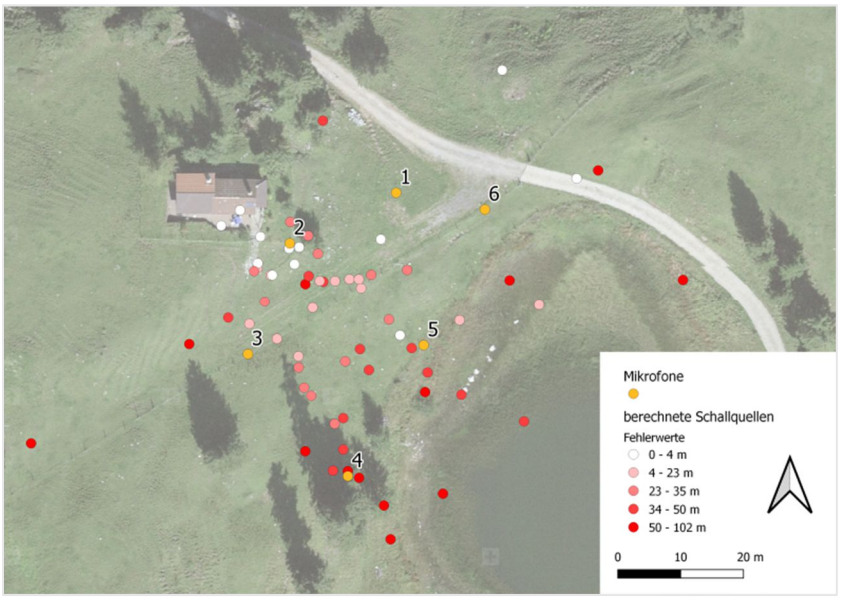


Abbildung 22: Resultate der Lokalisierung von 182 Rufen. Dargestellt sind 65 Punkte, die sich lokalisieren liessen. Davon haben 11 Punkte einen Fehlwert unter 4 m. Quelle: Bundesamt für Landestopographie swisstopo, Abbildung erstellt mit Qgis.

5 Diskussion

5.1 Automatische Auswertung

Allgemein zeigte sich eine gewisse Unschärfe bei der Beurteilung der Dauer von Rufreihen. Bei der manuellen Nachkontrolle der Resultate der Filtermethode (Schwellenwert 0.8) fanden sich Rufreihen mit einer Dauer von 67 min, die bei der manuellen Auswertung nicht markiert wurden. Diese Differenz kann bei kurzen Pausen zwischen Rufsequenzen und bei sehr undeutlichen Rufen entstehen. Dabei ist teilweise nicht ganz klar, ob es sich um eine Rufreihe handelt oder nicht. Die Resultate zur Sensitivität und Spezifität sind deshalb unter Berücksichtigung einer gewissen Unschärfe zu interpretieren.

5.1.1 Ishmael

Das Programm Ishmael konnte die gestellten Anforderungen nicht erfüllen. Bei den Methoden „Energy sum detection“ und „Spectrogram correlation“ stellten sich die Resultate als zu unspezifisch heraus. Die Methode „Matched filter“ fand vor allem laute Rufe sehr spezifisch. Jedoch funktionierte die Methode nur bei Rufen derselben Frequenz. Rufe, die nur leicht höher oder tiefer als die vorgegebene Frequenzkontur sind, fand das Programm nicht. Somit ist die automatische Auswertung von Langzeitaufnahmen zur Identifikation von Rufen der Geburtshelferkröten mit diesem Programm nicht möglich.

5.1.2 Kaleidoscope

Mit dem Einsatz eines fortgeschrittenen Klassifikators war es möglich, mit dem Programm Kaleidoscope deutliche Rufe automatisch zu erkennen. Rufer lassen sich bis zu einer Distanz von ca. 20 m zum Aufnahmegerät relativ zuverlässig erkennen. Es ist jedoch nicht möglich, eine Aufnahme anhand der gefundenen Rufe weiter auf nicht gefundene Rufe zu untersuchen. Weiter ist es auch nicht möglich, Rufe direkt zu Rufreihen zusammenzufügen. Mit Hilfe von R lassen sich die gefundenen Rufe zu Rufreihen kombinieren und als Raven-Selection-Table abspeichern. Dies wiederum ermöglicht es, die Resultate im Kontext der gesamten Aufnahme zu analysieren. Als grossen Vorteil ist die äusserst kurze Rechendauer zu unterstreichen, auch für sehr grosse Datenvolumen. Diese ermöglicht es, allenfalls bereits im Feld bei der Kontrolle der Geräte, einen kurzen Überblick über die Resultate zu erhalten. Um die Beschränkung auf eine relativ kleine abgedeckte Fläche von etwa 1 200 m^2 (20 m Radius) zu beheben, müssten die Position des Aufnahmegeräts nach einer gewissen Zeit verschoben werden. Dabei könnte es hilfreich sein, die Daten bereits im Feld zu analysieren und das Gerät anhand der Resultate neu zu platzieren.

Die geringe Anzahl Rufe, die das Programm in den Aufnahmen der Mehlbachgrube fand, ist wohl dem Zufall zuzuschreiben. Dies zeigt, dass negative Resultate dieser Methode keinesfalls als Negativnachweis interpretiert werden dürfen. Weiter zeigt sich, dass zur Erkennung von Rufen eine gewisse Lautstärke der Rufe gewährleistet sein muss. Während bei Aufnahmen aus nächster Nähe zur Schallquelle viele Methoden gut funktionieren (pers. Mitteilung D. Mennill 20.11.19), nimmt die Leistung mit der Distanz zur Schallquelle stark ab (Woelfel & McDonough, 2009). Da Langzeitaufnahmen typischerweise ein grösseres Gebiet abdecken ist dies das Hauptproblem der meisten Methoden zur automatischen Auswertung (Priyadarshani et al., 2018).

5.1.3 Filtermethode

In Kombination mit einer manuellen Nachkontrolle der Resultate lassen sich mit der Filtermethode auch leise Rufe aus bis zu 60 m Entfernung zuverlässig erkennen. Dies bedeutet beinahe eine Verzehnfachung der abgedeckten Fläche im Vergleich zu Kaleidoscope von schätzungsweise 1 200 auf etwa 11 300 m^2. Als Nachteil ist bei dieser Methode die lange Rechenzeit zu erwähnen. Da aber grosse Datenmengen in einem Batch-Prozess ausgewertet werden, ist es möglich, das Programm über Nacht ohne Überwachung laufen zu lassen. Zu Abstürzen des Systems ist es bei den Auswertungen nie gekommen. Hingegen ist eine Auswertung direkt im Feld nicht möglich. Der Aufwand zur Nachkontrolle der Resultate ist mit 15 % im Vergleich zur komplett manuellen Auswertung vertretbar.

Der Aufwand zur Nachkontrolle und der Ertrag an gefundenen Rufsequenzen hängen direkt vom Parameter „threshold.factor“ ab. Dieser bestimmt, ab welcher Lautstärke ein Geräusch ein Ruf sein kann. Wird der Schwellenwert zu hoch angesetzt, so bleiben viele Rufe unerkannt. Bei zu tiefem Schwellenwert wird der Aufwand zur Nachkontrolle unnötig hoch und die Rechendauer lang. Zudem können zu viele erkannte Signale dazu führen, dass eine Rufreihe nicht mehr als solche erkannt wird. Bei den Aufnahmen der Grube Mehlbach zeigte sich, dass ein Wert von 0.9, wie er für das Übungsfile optimal war, zu hoch ist. Bei einem Wert von 0.8 sind die Resultate bereits deutlich besser. Möglicherweise könnte der Wert noch weiter herabgesetzt werden. Bei Aufnahmereihen mit vielen Dateien macht es deshalb Sinn, diesen Wert zuerst an einem kleinen Teil der Aufnahmen zu testen.

Die Anwendung dieser Methode erfordert Grundkenntnisse in R. Mit der Erstellung einer Funktion ist die Handhabung aber so einfach wie möglich gestaltet und die Anleitung in Anhang XIII ist auf unerfahrene R-Benutzer ausgelegt.

Der Erfolg der Filtermethode zeigt, dass eine spezifische, auf ein Problem zugeschnittene Methode bessere Resultate liefern kann als ein professionelles Programm, das für eine breite Anwendung gedacht ist. Dies bedeutet aber auch, dass sich die Filtermethode nur sehr bedingt zur Auswertung von Lautäusserungen

anderer Arten eignet. Solche müssten sich ebenfalls durch kurze und regelmässig vorgetragene Rufe auszeichnen wie zum Beispiel die Gesänge von Sperlingskauz und Zwergohreule (Uthleb, 2012).

5.1.4 Empfehlungen zur Auswertung von Langzeitaufnahmen

Bevor eine Langzeitaufnahme geplant wird, muss definiert werden, ob die Untersuchung einen reinen Nachweis erbringen soll (inkl. einer Abschätzung des Bestands bei kleinen Vorkommen) oder, ob Daten zu Aktivität im Verlauf der Zeit interessieren.

Um die Aktivität von Rufern zu dokumentieren, sollten möglichst alle Rufsequenzen gefunden werden. Dazu ist die manuelle Auswertung mit RavenLite die genauste Methode der geprüften Möglichkeiten. Der Aufwand kann grob mit einer Minute pro Aufnahme-Stunde abgeschätzt werden (Quelle SA). Kaleidoscope könnte hier möglicherweise helfen einen Überblick zu erhalten, auf welchen Aufnahmedateien überhaupt deutliche Rufe vorhanden sind. Allerdings würden damit nur die deutlichen und nahe dem Aufnahmegerät abgegebenen Rufe erkannt.

Für einen reinen Anwesenheits-Nachweis empfiehlt es sich, das zu untersuchende Gebiet in ein Dreiecksnetz mit einer Kantenlänge von 35 m (maximale Distanz zu einem Eckpunkt ~20 m) einzuteilen. Entweder kann an jedem Eckpunkt ein Aufnahmegerät platziert werden oder ein einzelnes Gerät wird nach einer bestimmten Zeit zum nächsten Punkt verschoben. So lassen sich mögliche Rufe deutlich genug aufnehmen, um diese mit Kaleidoscope zu analysieren. Vor dem Umplatzieren können die Aufnahmen sogar gleich im Feld analysiert und dementsprechend die Aufnahmen weitergeführt oder bei einem gelungenen Nachweis abgebrochen werden. Kann Kaleidoscope keine Rufe finden, darf dies aber nicht als Negativnachweis deklariert werden. Ist ein Negativnachweis (die Aussage „es sind keine Rufer vorhanden“ im Gegensatz zur Aussage „es wurden keine Rufer gefunden“) aber von Interesse, so ist eine manuelle Auswertung zwingend nötig. Dies ist beispielsweise der Fall, wenn bei einem Bauvorhaben sichergestellt werden soll, dass kein Lebensraum der Geburtshelferkröte zerstört wird.

Ist es nicht möglich oder zu aufwändig, ein Gebiet mit mehreren Geräten oder mit einem Gerät an verschiedenen Standorten zu untersuchen, empfiehlt es sich, das Gerät an einer zentralen Stelle zu platzieren und die Aufnahmen in zwei Schritten zu analysieren. Zuerst kann mit Kaleidoscope schnell festgestellt werden, ob deutliche Rufe vorhanden sind. Falls dies nicht der Fall ist, hilft die Filtermethode auch undeutliche Rufe zu finden. Für einen Negativnachweis ist auch hier eine manuelle Auswertung nötig.

5.2 Lokalisierung

5.2.1 Laborversuch

Der Laborversuch zeigt, dass sich abgespielte Rufe von Geburtshelferkröten räumlich lokalisieren lassen und über eine Clusteranalyse auf die Anzahl Schallquellen zurückgeschlossen werden kann. Die Resultate sind weder sehr präzise (Standardabweichung) noch sehr akkurat (Distanz zur Schallquelle). Der Versuch mit mehreren Schallquellen lässt vermuten, dass beide Werte besser sind, je näher die Schallquelle beim Mittelpunkt der Aufnahmegeräte liegt. Zum selben Schluss kommen auch McGregor et al. und Mennill et al. (2010; 2012). Um rein auf die Anzahl Schallquellen zu schliessen, ist vor allem eine tiefe Standardabweichung wichtig, damit mehrere Schallquellen durch die Streuung nicht zu einer einzelnen verschmelzen.

5.2.2 Feldversuch

Der Feldversuch lieferte keine verlässlichen Resultate. Dies sowohl mit derselben Auswertungsmethode wie im Laborversuch, als auch mit diversen Anpassungen bei der Analyse, wie zum Beispiel der Auswertung von nur drei von sechs Aufnahmegeräten. Ein Problem stellt die hohe Anzahl Rufe pro Sekunde in den Aufnahmen dar. Da bei der Auswertung jeweils ein Zeitfenster von 0.4 s analysiert wird, fallen bei einer durchschnittlichen Rufrate von 3 Rufen pro Sekunde sehr häufig mehr als ein Ruf in ein Zeitfenster, was die Lokalisierung verfälscht. Geschieht dies nur selten, so spielt dies bei einer genügend hohen Anzahl

Rufe keine Rolle. Jedoch konnte ich auch Einzelrufe nicht lokalisieren. Auffällig ist der direkte Vergleich der sechs Aufnahmegeräte in Anhang XII. Während sich im Laborversuch die verschiedenen Aufnahmen bis auf leichte Unterschiede in der Lautstärke der Rufe recht ähnlich sahen, unterscheiden sich diese aus dem Feldversuch zum Teil sehr stark. So ist zum Beispiel in der Aufnahme des Geräts OW01 nur die Rufreihe eines einzelnen Individuums deutlich sichtbar, während das 18.7 m entfernte Gerät OW03 unzählige Rufreihen aufzeichnete. Einzelne Rufe lassen sich jedoch auf allen Aufnahmen zur selben Zeit finden, was zeigt, dass kein Fehler in der Synchronisierung vorliegt. Möglicherweise unterschieden sich die Rufe im Feldversuch von den abgespielten Rufen im Laborversuch. Wenn Rufer aus einem Versteck, wie zum Beispiel einem Mausloch heraus rufen, so kann dies zu einer Kanalisierung der Schallwellen führen. Dies könnte zu verschiedenen Reichweiten des Rufs in unterschiedlichen Richtungen führen. Hingegen konnten sich die Schallwellen aus dem Lautsprecher besser in alle Richtungen ausbreiten. Weiter müsste möglicherweise die tatsächliche Ruflautstärke überprüft werden, da die verwendete Lautstärke (84.6 dB) auf der Messung eines einzelnen Individuums basierte. Jedoch geben Márqez et al. (2016) mit 83.9 dB in 0.5 m Entfernung einen vergleichbaren Wert an.

Die grossen Unterschiede zwischen den Aufnahmen zeigen, dass die Abstände der Geräte nicht zu klein, sondern eher zu gross gewählt wurden. Das Untersuchungsgebiet mit einer Fläche von 915 m^2 deckt aber nur gerade 1.7 % des gesamten Amphibienlaichgebiets von nationaler Bedeutung ab (Abbildung 9). Rufende Geburtshelferkröten sind zudem auch ausserhalb des Gebiets zu finden. Eine Abdeckung des gesamten Gebiets oder des gesamten Bestands scheint also kaum umsetzbar. Mit dieser Methode wäre es demnach höchstens möglich, eine Stichprobefläche zu untersuchen und so einen Bestandsindex zu generieren.

5.2.3 Wichtige Erkenntnisse

Trotzt dem Scheitern der Methode im Feldversuch konnten wichtige Erkenntnisse für allfällige weiter Lokalisierungsversuche gesammelt werden.

Um die SM3 Aufnahmegeräte präzise zu synchronisieren, reicht eine Vorlaufzeit vor der Aufnahme von einer Minute, wie sie vorprogrammiert ist, häufig nicht. Dies obwohl das Gerät mit der Anzeige eines Dollarzeichens auf dem Display signalisiert, dass es ein GPS-Signal empfängt. Mit drei Minuten Vorlaufzeit (1 min Vorlauf, 1 min Aufnahme, 1 min Pause) kann dieses Problem meist behoben werden. Bei Verschiebungen über grosse Distanzen dauert die nötige Vorlaufzeit länger.

Die vom Aufnahmegerät abgespeicherten, wie auch die mit einem handelsüblichen GPS-Gerät ermittelten Koordinaten, sind zur Lokalisierung der Schallquellen nicht genügend akkurat. Von zwei Fixpunkten aus lassen sich die Standorte der Aufnahmegeräte mit einem Lasermessgerät aber genügend genau ausmessen.

Der Knackpunkt der Lokalisierung stellt die präzise Messung der Eintreff-Zeitdifferenzen dar. Wilson et al. (Quelle) führen dies mit einer Kreuzkorrelation im Programm RavenPro durch. Da mir das kostenpflichtige Programm nicht zur Verfügung stand, führte ich eine Kreuzkorrelation in R (Funktion ccf) durch. Die Resultate waren jedoch nicht genügend genau. Ob dies an der Methode oder am Ruftyp liegt, ist nicht klar. Kurze und sehr einfach aufgebaute Rufe wie der der Geburtshelferkröte sind zur Lokalisierung weniger geeignet als zum Beispiel eine komplexe Strophe eines Singvogels (Mennill et al., 2012). Mit der angewandten Methode fand ich jedoch eine Lösung, die zumindest im Laborversuch vertretbare Resultate lieferte.

5.3 Fazit

Diese Arbeit zeigt auf, dass es möglich ist, Langzeitaufnahmen von Geburtshelferkröten automatisiert auszuwerten. Das Programm Kaleidoscope eignet sich, um deutliche Rufe sehr effizient zu finden. In einer Aufnahme mit deutlichen Rufen fand das Programm 88.8 % der Rufsequenzen. Dies ohne manuelle Nachkontrolle und bei sehr kurzer Rechendauer. Undeutliche Rufe findet das Programm aber kaum. Sol-

che lassen sich hingegen mit der Filtermethode erkennen. Mit einer manuellen Nachkontrolle fand das Programm in einer Aufnahme mit undeutlichen Rufen 73.4 % der Rufsequenzen bei einem zeitlichen Aufwand von 15 %, verglichen mit der manuellen Auswertung. Die beiden Methoden lassen sich auch kombinieren.

Weiter zeigte sich, dass es unter Laborbedingungen mit abgespielten Geburtshelferkröten-Rufen möglich ist, Schallquellen mit Hilfe von synchronisierten Tonaufnahmen räumlich zu verorten und so auf deren Anzahl zu schliessen. Im Feldversuch scheiterte die Methode jedoch. Auch bei Behebung der technischen Probleme würde sich die Methode wohl nicht eignen, um ein herkömmliches Monitoring zu ersetzten. Höchstens eine Stichprobefläche könnte untersucht werden.

5.4 Ausblick

Die Lokalisierung von Schallquellen ist zwar nicht trivial, bietet aber spannende Anwendungsmöglichkeiten. Trotz dem Misserfolg bei der Lokalisierung von Geburtshelferkröten, sollte die Methode weiterverfolgt und bei anderen Tierarten getestet und angewendet werden. Bei Arten mit lauteren und komplexer aufgebauten Vokalisationen könnte die Lokalisierung entsprechend einfacher sein.

Mit den hier aufgezeigten Möglichkeiten zur automatischen Auswertung von Langzeitaufnahmen, soll es in Zukunft möglich sein, Vorkommen von Geburtshelferkröten einfacher und effizienter nachzuweisen. Dies könnte dazu beitragen, unbekannte Vorkommen zu entdecken und den Status von sehr kleinen oder verschollenen Vorkommen abzuklären. Das Wissen über den Zustand von Populationen ist für deren Schutz unerlässlich.

6 Literaturverzeichnis

Bardeli, R., Wolff, D., Kurth, F., Koch, M., Tauchert, K.-H., & Frommolt, K.-H. (2010). Detecting bird sounds in a complex acoustic environment and application to bioacoustic monitoring. *Pattern Recognition Letters*, *31*(12), 1524–1534. https://doi.org/10.1016/j.patrec.2009.09.014

Borgula, A., & Zumbach, S. (2003). Verbreitung und Gefährdung der Geburtshelferkröte (Alytes obstetricans) in der Schweiz. *Zeitschrift für Feldherpetologie*, *10*, 11–26.

Bundesamt für Naturschutz BfN (Hrsg.). (2017). *Bewertungsschemata für die Bewertung des Erhaltungsgrades von Arten und Lebensraumtypen als Grundlage für ein bundesweites FFH-Monitoring.* https://www.bfn.de/fileadmin/BfN/service/Dokumente/skripten/Skript480.pdf

Bundesamt für Umwelt BAFU (Hrsg.). (2005). *Rote Liste der gefährdeten Arten der Schweiz: Amphibien*. Bundesamt für Umwelt BAFU.

Bundesamt für Umwelt BAFU (Hrsg.). (2017). *Bundesinventar der Amphibienlaichgebiete von nationaler Bedeutung, Objektbaltt 35*. https://data.geo.admin.ch/ch.bafu.bundesinventare-amphibien/objectsheets/2017revision/ow35.pdf

Burkhalter, F. (2019). *Automatisierte Erkennung der Balzaktivität von Birkhähnen (Tetrao tetrix) in R anhand bioakustischer Aufnahmen. Bachelorarbeit.* Zürcher Hochschule für angewandte Wissenschaften ZHAW.

CIMRS Bioacoustics Lab. (o. J.). *Exploring automatich detection with Ishmael*. Abgerufen 15. Mai 2020, von http://bioacoustics.us/ishmael.html

Clink, D. J., & Klinck, H. (2019). *GIBBONR: An R package for the detection and classification of acoustic signals using machine learning*.

Cornell Lab of Ornithology. (o. J.). *Raven Lite*. Abgerufen 15. Dezember 2020, von http://ravensoundsoftware.com/software/raven-lite/

Garrido Sanchis, A., Bertolelli, L., Maria Hoefer, A., Yebra Alvarez, M., & Munasinghe, K. (2020). The FrogPhone: A novel device for real-time frog call monitoring. *Methods in Ecology and Evolution*, *11*(2), 222–228. https://doi.org/10.1111/2041-210X.13332

Hachtel, M., Schlüpmann, M., Thiesmeier, B., & Weddeling, K. (2009). Methoden der Amphibienerfassung—Eine Übersicht. *Zeitschrift für Feldherpetologie*, *15*, 7–84.

Hofer, U. (2016). *Evidenzbasierter Artenschutz. Bergiffe, Konzepte, Methoden*. Haupt Verlag.

Jakober, M. (2019). Geburtshelferkröte: Wie ein Vorkommen retten? *Inside*, *2/19*, 38–42.

Knight, E., Hannah, K., Foley, G., Scott, C., Brigham, R., & Bayne, E. (2017). Recommendations for acoustic recognizer performance assessment with application to five common automated signal recognition programs. *Avian Conservation and Ecology*, *12*. https://doi.org/10.5751/ACE-01114-120214

Köhler, J., Jansen, M., Rodriguez, A., Kok, P., Toledo, L. F., Emmrich, M., Glaw, F., Haddad, C., Rödel, M.-O., & Vences Miguel. (2017). The use of bioacoustics in anuran taxonomy: Theory, terminology, methods and recommendations for best practice. *Zootaxa*, *4251*, 1–124. https://doi.org/10.11646/zootaxa.4251.1.1

Lalkhen, A. G., & McCluskey, A. (2008). Clinical tests: Sensitivity and specificity. *Continuing Education in Anaesthesia Critical Care & Pain*, *8*(6), 221–223. https://doi.org/10.1093/bjaceaccp/mkn041

Márquez, R., Bosch, J., & Penna, M. (2016). Sound pressure level of advertisement calls of Alytes cisternasii and Alytes obstetricans (Anura, Discoglossidae). *Bioacoustics*, *16(1)*, 27–37. https://doi.org/10.1080/09524622.2006.9753562

Marti, P. (2019). *Nachweis von Geburtshelferkröten (Alytes obstetricans) mit Hilfe von Bioakustik. Semesterarbeit.* Zürcher Hochschule für angewandte Wissenschaften ZHAW.

Mcgregor, P., Dabelsteen, T., Clark, C., Bower, J. L., & Holland, J. (2010). Accuracy of a passive acoustic location system: Empirical studies in terrestrial habitats. *Ethology Ecology & Evolution*, *July 1997*, 269–286. https://doi.org/10.1080/08927014.1997.9522887

Mennill, D. J., Battiston, M., Wilson, D. R., Foote, J. R., & Doucet, S. M. (2012). Field test of an affordable, portable, wireless microphone array for spatial monitoring of animal ecology and behaviour. *Methods in Ecology and Evolution*, *3*(4), 704–712. https://doi.org/10.1111/j.2041-210X.2012.00209.x

Meyer, A., Zumbach, S., Schmidt, B., & Monney, J.-C. (2014). *Auf Schlangenspuren und Krötenpfaden Amphibien und Reptilien der Schweiz*. Haupt Verlag.

Naturforschende Gesellschaft Obwalden und Nidwalden NAGON (Hrsg.). (2001). *Amphibien und Reptilien in Ob- und Nidwalden. Mit Beiträgen aus den Naturwissenschaften, der Sagenwelt und der Volksmedizin.*

Obrist, M. K., Pavan, G., Sueur, J., Riede, K., Llusia, D., & Márquez, R. (2010). Bioacoustics approaches in biodiversity inventories. *Abc Taxa*, *8*, 68–99.

Pamguard. (o. J.). *A User's Introduction to Pamguard*. Abgerufen 15. Mai 2020, von http://www.pamguard.org/42_UserTutorial.html

Priyadarshani, N., Marsland, S., & Castro, I. (2018). Automated birdsong recognition in complex acoustic environments: A review. *Journal of Avian Biology*, *49*(5), jav-01447. https://doi.org/10.1111/jav.01447

Rhinehart, T. A., Chronister, L. M., Devlin, T., & Kitzes, J. (2020). Acoustic localization of terrestrial wildlife: Current practices and future opportunities. *Ecology and Evolution*, *10*(13), 6794–6818. https://doi.org/10.1002/ece3.6216

Robin, K., Graf, R. F., & Schnidrig, R. (2017). *Wildtiermanagement—Eine Einführung*. Haupt Verlag.

Schlup, B., Bühler, C., Stickelberger, C., & Walker, R. (2018). Neues Tool für Erfolgskontrollen—Amphibien. Programm zum Wiedererkennen von Individuen mit Punktemustern. *Inside*, *1*, 20–23.

Sueur, J. (2018). *Sound Analysis and Synthesis with R*. Springer International Publishing. https://doi.org/10.1007/978-3-319-77647-7

Uthleb, H. (2012). *Die Geburtshelferkröte. Brutpflege ist männlich*. Laurenti-Verlag.

Uthleb, H. (2020). Der Zusammenbruch einer Population der Nördlichen Geburtshelferkröte (Alytes obstetrichans). *RANA*, *20*, 48–50.

Wildliefe Acoustics. (o. J.). *Kaleidoscope Pro 5. User Guide*. Abgerufen 13. Mai 2020, von https://www.wildlifeacoustics.com/products/kaleidoscope

Wilson, D., Battiston, M., Brzustowski, J., & Mennill, D. (2013). Sound Finder: A new software approach for localizing animals recorded with a microphone array. *Bioacoustics*, *23*. https://doi.org/10.1080/09524622.2013.827588

Woelfel, M., & McDonough, J. (2009). *Distant Speech Recognition*. Wiley.

Anhang

Anhangsverzeichnis

Anhang I

R-Skript zum Exportieren von Resultaten von Ishmael nach Raven

```
# set working directory
setwd("C:/Users/pmar/Desktop/Ishmael")

# a selection table is created and can be laid over a spectrogram in Raven
# open the output file in Raven with Open Selection Table

# read in data
data <- read.table("logfile.txt", sep = ",", header = TRUE)


# define the length of the selection table
length <- length(data$Annotation.name)

# column names
Header <-
  c(
    "Selection",
    "View",
    "Channel",
    "Begin Time (s)",
    "End Time (s)",
    "Low freq (Hz)",
    "High freq (Hz)",
    "Delta Time (s)",
    "Delta freq (Hz)",
    "Avg Power Density (dB FS)",
    "Annotation"
  )

# add values
Selection <- seq(from = 1, to = length, by = 1)
View <- rep("Spectrogram 1", length = length)
Channel <- rep(1, length = length)
BeginTime <- as.numeric(data$Start.Time-0.25)
EndTime <- as.numeric(data$End.Time+0.25)
Lowfreq <- 1000
Highfreq <- 1500
DeltaTime <- rep("", length = length)
Deltafreq <- rep("", length = length)
AvgPowerDensity <- rep("", length = length)
Annotation <- rep("", length = length)

# combine to data.frame
input_raven <-
  data.frame(
    Selection,
    View,
    Channel,
    BeginTime,
    EndTime,
```

```
    Lowfreq,
    Highfreq,
    DeltaTime,
    Deltafreq,
    AvgPowerDensity,
    Annotation
  )

names(input_raven) <- Header

# write output file
write.table(
  input_raven,
  "input_raven_Ishmael.txt",
  row.names = FALSE,
  dec = ".",
  sep = "\t",
  quote = FALSE
)
```

Anhang II

R-Skript zum Exportieren von Resultaten von Kaleidoscope nach Raven

```
# set working directory
setwd("C:/Users/pmar/Desktop/Kaleidoscope")

# a selection table is created and can be laid over a spectrogram in Raven
# open the output file in Raven with Open Selection Table

# read in data
data<-read.delim("cluster.csv",sep=";")


# List of input files
in.File.list<-as.character(unique(data$IN.FILE))


for(i in 1:length(in.File.list)){
  data.subset<-subset(data,data$IN.FILE==in.File.list[i])

  # define the length of the selection table
  length <- length(data.subset$INDIR)

  # column names
  Header <-
    c(
      "Selection",
      "View",
      "Channel",
      "Begin Time (s)",
      "End Time (s)",
      "Low freq (Hz)",
      "High freq (Hz)",
      "Annotation"
    )

  # add values
  Selection<- seq(from=1,to=length,by=1)
  View<- rep("Spectrogram 1",length = length)
  Channel<-rep(1,length = length)
  BeginTime<-data.subset$OFFSET-0.5
  EndTime<-data.subset$OFFSET+data.subset$DURATION+0.5
  Lowfreq<-rep(1100,length = length)
  Highfreq<-rep(1350,length = length)
  DeltaTime<- rep("",length = length)
  Deltafreq<- rep("",length = length)
  AvgPowerDensity<- rep("",length = length)
  Annotation<- rep(data.subset$TOP1MATCH.)

  # combine to data.frame
  input_raven <-
    data.frame(
      Selection,
```

```
      View,
      Channel,
      BeginTime,
      EndTime,
      Lowfreq,
      Highfreq,
      DeltaTime,
      Deltafreq,
      AvgPowerDensity,
      Annotation
    )

  names(input_raven) <- Header


  # write output file
  write.table(
    input_raven,
    paste(in.File.list[i],"_input_raven",".txt",sep=""),
    row.names = FALSE,
    dec=".",
    sep="\t",
    quote=FALSE)
}
```

Anhang III

R-Skript zur Berechnung der Wahrheitsmatrix inkl. Spezifität und Sensitivität

```
#Aufnahmedauer in s
dur<-6*60*60


library(seewave)
library(tuneR)

#forloop
setwd("C:/Users/pmar/OneDrive - ZHAW/Studium/6. Semester/BA/Detector/matrix
/manuell")

list_of_files <- list.files(pattern = ".txt")

matrix.vector<-c(0,0,0,0)

for(x in 1:length(list_of_files)){

#manual data
setwd("C:/Users/pmar/OneDrive - ZHAW/Studium/6. Semester/BA/Detector/matrix
/manuell")

list_of_files <- list.files(pattern = ".txt")


data.manual<-read.table(paste(list_of_files[x]),sep="\t",header = TRUE)

#auto data
setwd("C:/Users/pmar/OneDrive - ZHAW/Studium/6. Semester/BA/Detector/soundf
iles/output")

list_of_files <- list.files(pattern = ".txt")

data.automatic<-read.table(paste(list_of_files[x]),sep="\t",header = TRUE)


time.seq<-seq(1:dur)

on.off.automatic<-NA
for(i in 1:dur){
  on.off.automatic.l<-ifelse(i<=data.automatic$End.Time..s.[max(which(data.
automatic$Begin.Time..s.<=i))],1,0)
  on.off.automatic<-c(on.off.automatic,on.off.automatic.l)
}
on.off.automatic<-on.off.automatic[-1]
on.off.automatic[is.na(on.off.automatic)]<-0
on.off.automatic

on.off.manual<-NA
for(i in 1:dur){
  on.off.manual.l<-ifelse(i<=data.manual$End.Time..s.[max(which(data.manual
```

```
$Begin.Time..s.<=i))],1,0)
  on.off.manual<-c(on.off.manual,on.off.manual.l)
}
on.off.manual<-on.off.manual[-1]
on.off.manual[is.na(on.off.manual)]<-0
on.off.manual




matrix<-NA
for(i in 1:length(on.off.automatic)){
  matrix.l<-ifelse(on.off.automatic[i]==0&&on.off.manual[i]==0,"TN",
                   ifelse(on.off.automatic[i]==1&&on.off.manual[i]==0,"FP",
                          ifelse(on.off.automatic[i]==0&&on.off.manual[i]==
1,"FN",
                                 ifelse(on.off.automatic[i]==1&&on.off.manu
al[i]==1,"TP",))))
  matrix<-c(matrix,matrix.l)
}
matrix<-matrix[-1]
matrix

tm<-table(matrix)
true.rate<-sum(as.numeric(tm["TP"]),as.numeric(tm["TN"]))/dur*100
FP.rate<-as.numeric(tm["FP"])/dur*100
FN.rate<-as.numeric(tm["FN"])/dur*100

r1<-c("","Ground truth positiv","Ground truth negativ")
r2<-c("System true",as.numeric(tm["TP"]),as.numeric(tm["TN"]))
r3<-c("System false",as.numeric(tm["FP"]),as.numeric(tm["FN"]))
r4<-c(true.rate,FP.rate,FN.rate)


truth.matrix<-as.data.frame(rbind(r1,r2,r3,r4))

wave.name<-sapply(strsplit(list_of_files[x],"\\."),"[[",1)
wave.name<-sapply(strsplit(wave.name,"/"),"[[",1)

setwd("C:/Users/pmar/OneDrive - ZHAW/Studium/6. Semester/BA/Detector/matrix
")
write.table(truth.matrix,paste(wave.name,"_truth_matrix.csv"),sep=";",row.n
ames = FALSE,col.names = FALSE)

matrix.vector.l<-c(as.numeric(tm["TP"]),as.numeric(tm["TN"]),as.numeric(tm[
"FP"]),as.numeric(tm["FN"]))
matrix.vector.l[is.na(matrix.vector.l)]<-0
matrix.vector<-matrix.vector+matrix.vector.l
}
matrix.vector


r1<-c("","Ground truth positiv","Ground truth negativ")
r2<-c("System true",matrix.vector[1],matrix.vector[2])
```

```
r3<-c("System false",matrix.vector[3],matrix.vector[4])

true.rate<-sum(matrix.vector[1],matrix.vector[2])/(dur*length(list_of_files
))*100
FP.rate<-matrix.vector[3]/(dur*length(list_of_files))*100
FN.rate<-matrix.vector[4]/(dur*length(list_of_files))*100

r4<-c(true.rate,FP.rate,FN.rate)


truth.matrix<-as.data.frame(rbind(r1,r2,r3,r4))
truth.matrix
write.table(truth.matrix,"total_truth_matrix1.csv",sep=";",row.names = FALS
E,col.names = FALSE)
```

Anhang IV

R-Skript zum Erstellen eines Linerangeplots

```
library(ggplot2)
library(tidyverse)
library(lubridate)
library(ggpubr)
library(cowplot)

#automatisch
setwd("C:/Users/pmar/OneDrive - ZHAW/Studium/6. Semester/BA/Kaleidoscope/Auswertung_Kestenholz_Dickbangrube/output neu/call seq")

data<-read.delim("Kombinierte_Liste_neu.csv",sep=";")
auto=data.frame(datum=data$date1, lower=data$real.time.begin, upper=data$real.time.end)


#manuell
setwd("C:/Users/pmar/OneDrive - ZHAW/Studium/6. Semester/BA/Kaleidoscope/Auswertung_Kestenholz_Dickbangrube/manuell")

data<-read.delim("Kombinierte_Liste_neu.csv",sep=";")
manual=data.frame(datum=data$date1, lower=data$real.time.begin, upper=data$real.time.end)



#sunset
ss.time<-unique(data$time1)
ss.data=data.frame(datum=unique(manual$datum), lower=c(ss.time,ss.time+0.125), upper=c(ss.time+0.0002,ss.time+0.1252))
#ss.data<-ss.data[order(ss.data$datum,decreasing = TRUE),]


colors <- c("automatsich"="orange","manuell"="blue","Aufnahmestart und -ende"="black")

setwd("C:/Users/pmar/OneDrive - ZHAW/Studium/6. Semester/BA/Arbeit/Abbildungen")

jpeg("Lineage_Dickan_Kaleidoscope.jpeg",width = 1800,height = 800,pointsize = 20,res = 200,quality = 100)
ggplot() +
  geom_linerange(data=auto, mapping=aes(x=datum, ymin=upper, ymax=lower), size=6, color="orange")+
  geom_linerange(data=manual, mapping=aes(x=datum, ymin=upper, ymax=lower), size=2, color="blue")+
  geom_linerange(data=ss.data, mapping=aes(x=datum, ymin=upper, ymax=lower), size=6, color="black")+
  coord_flip()+
  labs(y="Zeit [h]",
       x="")+
```

```
  scale_y_time(labels=c("19:00","20:00","21:00","22:00"),breaks = c(19/24,2
0/24,21/24,22/24)) +
  geom_line(aes(y = auto$datum[1],1, color = "automatsich"), size = 2) +
  geom_line(aes(y = auto$datum[1],1, color = "manuell"), size = 2) +
  geom_line(aes(y = auto$datum[1],1, color = "Aufnahmestart
  und -ende"), size = 2) +
  labs(x = "Datum",
       y = "Uhrzeit",
       color = "") +
  scale_color_manual(values = colors)
dev.off()
```

Anhang V

R-Skript um Einzelrufe aus Kaleidoscope in Rufreihen umzuwandeln

```
setwd("C:/Users/pmar/OneDrive - ZHAW/Studium/6. Semester/BA/Kaleidoscope/Auswertung_Kestenholz_Dickbangrube/output neu/cluster")

max.call.int.length<-40
min.callseq.length<-6

data<-read.delim("cluster.csv",sep=";")
#View(data)

setwd("C:/Users/pmar/OneDrive - ZHAW/Studium/6. Semester/BA/Kaleidoscope/Auswertung_Kestenholz_Dickbangrube/output neu/call seq")

data<-data[order(data$OFFSET),]
in.File.list<-as.character(unique(data$IN.FILE))
in.File.list<-in.File.list[-which(in.File.list=="")]
#in.File.list<-in.File.list[-which(in.File.list=="")]

#export to raven####
for(i in 1:length(in.File.list)){

  #Subset file und alob
  data.subset<-subset(data,data$IN.FILE==in.File.list[i])
  data.subset<-subset(data.subset,data.subset$TOP1MATCH.=="ALOB")

  #Zeitdifferenz zwischen den einzelnen Rufen
  dif<-NA
  for(x in 1:length(data.subset$OFFSET)){
    dif.l<-data.subset$OFFSET[x+1]-data.subset$OFFSET[x]
    dif<-c(dif,dif.l)
  }
  dif<-dif[-1]


  #Start einer neuen Sequenz
  start.callseq<-which(dif>=max.call.int.length)+1

  start<-1
  for(a in 2:length(dif)){
    start.l<-ifelse(is.element(a,start.callseq),1,2)
    start<-c(start,start.l)
  }

  #Id fur jede Sequenz
  callseq.id<-1
  for(b in 2:length(dif)){
    callseq.id.l<-ifelse(start[b]==1,callseq.id[b-1]+1,callseq.id[b-1])
    callseq.id<-c(callseq.id,callseq.id.l)
  }

  #Mindestlange einer Sequenz
```

```
  callseq<-unique(which(table(callseq.id)>=min.callseq.length))

  #Start und Endzeit jeder Sequenz

  callseq.start<-NA
  callseq.end<-NA
  for(c in callseq){
    callseq.start.l<-min(data.subset$OFFSET[which(callseq.id==c)])
    callseq.start<-c(callseq.start,callseq.start.l)
    callseq.end.l<-max(data.subset$OFFSET[which(callseq.id==c)])
    callseq.end<-c(callseq.end,callseq.end.l)
  }
  callseq.start<-callseq.start[-1]
  callseq.end<-callseq.end[-1]

  #export to Raven
  length<-ifelse(length(callseq.start)>0,length(callseq.start),0)

  Header<-c("Selection","View","Channel","Begin Time (s)","End Time (s)","L
ow freq (Hz)","High freq (Hz)","Delta Time (s)","Delta freq (Hz)","Avg Powe
r Density (dB FS)","Annotation")
  Selection<- seq(from=1,to=length,by=1)
  View<- rep("Spectrogram 1",length = length)
  Channel<-rep(1,length = length)
  BeginTime<-callseq.start
  EndTime<-callseq.end
  Lowfreq<-rep(1100,length = length)
  Highfreq<-rep(1350,length = length)
  DeltaTime<- rep("",length = length)
  Deltafreq<- rep("",length = length)
  AvgPowerDensity<- rep("",length = length)
  Annotation<- rep("",length)

  input_raven<-data.frame(Selection,View,Channel,BeginTime,EndTime,Lowfreq,
Highfreq,DeltaTime,Deltafreq,AvgPowerDensity,Annotation)

  names(input_raven)<-Header

  # set file name, default "input_raven_warbler.txt"
  write.table(input_raven,paste(in.File.list[i],"_callseq_input_raven",".tx
t",sep=""),row.names = FALSE,dec=".",sep="\t",quote=FALSE)

}
```

Anhang VI

R-Skript zur Funktion “find.alob”

```
find.alob<-function(
  # input audio channel (1 or 2)
  channel=1,
  # length of section processed at once in minutes
  section.length.min=2,
  # factor to set threshold (lower -> more detects)
  threshold.factor=0.8,
  # maximal cluster length in time windows
  max.cluster.length=3,
  # maximal cluster height in frequency windows
  max.cluster.height=6,
  # minimal cluster height in frequency windows
  min.cluster.height=2,
  # time difference between calls in s
  threshold.fix.low=1,
  threshold.fix.high=20,
  # factor +/- median of time difference between calls
  median.threshold.low=0.6,
  median.threshold.high=1.6,
  # minimal length of a call sequence
  min.seq.length=5,

  # output
  # raven selection table of all calls detected (unfiltered)
  calls.detected=FALSE,
  # raven selection table of call sequences as lines
  lines.call.seq=FALSE,
  # raven selection table of call sequences as box
  call.seq=TRUE
){


#library####
  if(!require(seewave)){
    install.packages("seewave")}
  if(!require(tuneR)){
    install.packages("tuneR")}
  if(!require(tidyverse)){
    install.packages("tidyverse")}
  if(!require(plyr)){
    install.packages("plyr")}
  if(!require(fftw)){
    install.packages("fftws")}
  if(!require(pkgcond)){
    install.packages("pkgcond")}

library(seewave)
library(tuneR)
library(tidyverse)
library(plyr)
```

```
library(fftw)
library(pkgcond)
#####

#Start time
t1<-Sys.time()

wd<-getwd()
list.f<-list.files(pattern=".wav")

#batch
for(z in 1:length(list.f)){


  #choose sound file####
  setwd(wd)
  wav = readWave(paste(list.f[z]),header=TRUE)
  section.length<-wav$sample.rate*60*section.length.min
  n.section<-ceiling(wav$samples/section.length)
  #####


  df.call.seq.comp<-NA
  df.end.comp<-NA
  for(s in 0:(n.section-1)){
    wav.new<-readWave(paste(list.f[z]),from = s*section.length,to = (s+1)*s
ection.length)
    wav.new<-downsample(wav.new,8000)
    # progress
    print(paste(list.f[z],round(s/(n.section-1)*100,1),"%"))

    #Parameter####
    window.length<-512
    time.dif<-1/wav.new@samp.rate*window.length
    freq.dif<-wav.new@samp.rate/window.length/1000
    cluster.times=max.cluster.length+1
    #####


    #Fouriertransformation####

    ss<-spectro(wav.new, plot = F,fftw=TRUE,wl=window.length,channel = chan
nel)
    sp<-as.data.frame(ss$amp)
    colnames(sp)<-ss$time
    rownames(sp)<-ss$frq

    spf<-sp %>% slice(97:65)

    rownames(spf)<-ss$frq[97:65]

    #View(spf)

    #write.table(spf,file="spectrogram.csv",sep = ";",dec=".")
```

```
#####


#detection of values above threshold####
spf<--(spf)

#thresholds
median<-median(as.matrix(spf))
sd<-sd(as.matrix(spf))

threshold<-median-2*sd*threshold.factor


detects<-as.data.frame(which(spf<=threshold,arr.ind=TRUE))

if(is_empty(detects$row)){
  df.end<-as.data.frame(cbind("cluster.name"=0,
                "start.time"=0,
                "end.time"=0,
                "low.freq"=0,
                "high.freq"=0,
                "height.cluster"=0,
                "length.cluster"=0))
  }else{
"time(s)"<-as.vector(detects$col*time.dif)
frq<-as.vector(detects$row)
frq<-frq-33
frq<-frq*-1
frq<-(frq*freq.dif)+1
#####


#cluster vertical####

c.duplicates<-duplicated(`time(s)`)*1
n_occur<-data.frame(table(`time(s)`))


n<-length(`time(s)`)
anz.dup<-rep(NA,n)
pl.oc<-rep(NA,n)

if(is.na(`time(s)`[1])){}else{
  for(i in c(1:n)){
    pl.oc[i]<-which(`time(s)`[i]==n_occur)
    anz.dup[i]<-n_occur$Freq[pl.oc[i]]
  }
}


df.detects<-as.data.frame(cbind(`time(s)`,frq,anz.dup))


n<-length(n_occur$time.s.)
```

```
    df=data.frame(matrix(ncol=33,nrow=n))


    for(i in 1:n){
      a<-rep(NA,length=33)

      clu.freq<-df.detects$frq[c(which(df.detects$`time(s)`==  n_occur$time
.s.[i]),a)]

      clu.freq<-clu.freq[1:33]

      df[i,]<-clu.freq
    }


    #df.neu mit widerholter start freq
    n<-length(df$X1)

    df.neu=data.frame(matrix(ncol=33,nrow=n))

    df.neu[1]<-df[1]

    for(i in 2:33){
      df.neu[i]<-ifelse(df[i-1]==df[i]+freq.dif,
                        df.neu[[i-1]],
                        df[[i]])
    }


    #df.cluster.time freq(Var1) und anz(freq)
    df.cluster.time<-data.frame(table(as.numeric(as.vector(df.neu[1,]))))

    n<-length(df.neu$X1)

    if(n==1){}else{
      for(i in 2:n){
        cluster<-data.frame(table(as.numeric(as.vector(df.neu[i,]))))
        df.cluster.time<-rbind(df.cluster.time,cluster)
      }
    }



    #df.cluster.time vertical cluster
    n<-length(df.cluster.time$Var1)

    time.cluster<-rep(NA,n)
    time.cluster[1]<-`time(s)`[1]
    b<-df.cluster.time$Freq[1]

    for(i in 2:n){
      b<-b+df.cluster.time$Freq[i]
      time.cluster[i]<-`time(s)`[b]
    }
```

```
    if(is.na(df.cluster.time[1,])){
      df.cluster.time<-as.data.frame(cbind(0,0))
      colnames(df.cluster.time)<-c("Var1","Freq")
    }else{
      df.cluster.time<-cbind(df.cluster.time,time.cluster)
    }


    suppressMessages(df.cluster.time<-rename(df.cluster.time, c("Var1"="Hig
hfreq", "Freq"="size.cluster")))


    Lowfreq<-as.numeric(levels(df.cluster.time$Highfreq))[df.cluster.time$H
ighfreq]-((df.cluster.time$size.cluster-1)*freq.dif)

    if(is.na(Lowfreq[1])){}else{
      df.cluster.time<-cbind(df.cluster.time,Lowfreq)
    }
    #####


    #cluster horizontal####

    #look for cluster in next time window
    n<-length(df.cluster.time$time.cluster)

    same.cluster<-c(NA)

    for(i in 1:n){
      same.cluster[i]<-is.element(round(time.cluster[i]+time.dif,digits = 3
),round(time.cluster,digits = 3))
    }

    #check if two cluster overlap
    ifelse(is.null(df.cluster.time$time.cluster[1]),to<-1,to<-length(df.clu
ster.time$time.cluster))
    pl.cl<-seq(from=1,to=to,by=1)

    n=length(pl.cl)

    #lowfreq1

    Lowfrq1<-c(NA)

    for(i in pl.cl){
      Lowfrq1[i]<-df.cluster.time$Lowfreq[i]
    }
    Lowfrq1<-Lowfrq1[!is.na(Lowfrq1)]

    #Highfreq1+2

    Highfreq<-as.numeric(levels(df.cluster.time$Highfreq))[df.cluster.time$
```

```
Highfreq]

    #Highfreq1
    Highfrq1<-c(NA)

    for(i in pl.cl){
      Highfrq1[i]<-Highfreq[i]
    }
    Highfrq1<-Highfrq1[!is.na(Highfrq1)]


    df.freq.cluster.true<-data.frame(matrix(ncol=5,nrow=length(pl.cl)))



    for(x in 1:10){
      #lowfreq2
      Lowfrq2<-c(NA)

      for(i in pl.cl){
        Lowfrq2[i]<-df.cluster.time$Lowfreq[i+x]
      }
      Lowfrq2<-Lowfrq2[!is.na(Lowfrq2)]
      Lowfrq2

      #Highfreq2
      Highfrq2<-c(NA)

      for(i in pl.cl){
        Highfrq2[i]<-Highfreq[i+x]
      }
      Highfrq2<-Highfrq2[!is.na(Highfrq2)]
      Highfrq2


      n<-length(Lowfrq1)

      frq.cluster.true<-c(NA)



      for(i in 1:n){
        frq.cluster.true[i]<-
          ifelse(
            (Lowfrq1[i]==Lowfrq2[i]||
               Lowfrq1[i]==Highfrq2[i]||
               Highfrq1[i]==Lowfrq2[i]||
               Highfrq1[i]==Highfrq2[i]||
               Highfrq1[i]>=Highfrq2[i]&&
               Lowfrq1[i]<=Highfrq2[i]||
               Highfrq1[i]<=Highfrq2[i]&&
               Highfrq1[i]>=Lowfrq2[i])
          ,1,NA)
      }
```

```
      frq.cluster.true

      for(i in 1:n){
        frq.cluster.true[i]<-
          ifelse(frq.cluster.true[i]==1&&
                   round(df.cluster.time$time.cluster[i]+time.dif,digits =
3)==
                   round(df.cluster.time$time.cluster[i+x],digits = 3),1,NA
)
      }

      frq.cluster.true


      df.freq.cluster.true[,x]<-frq.cluster.true
    }

    #look for start time of cluster
    true.cluster.start.position<-pl.cl*df.freq.cluster.true
    true.cluster.start.position<-true.cluster.start.position[!is.na(true.cl
uster.start.position)]
    true.cluster.start.position<-sort(true.cluster.start.position,decreasin
g = FALSE)

    for(i in 1:10){
      df.freq.cluster.true[,i]<-df.freq.cluster.true[i]*i
    }


    #true cluster end position
    true.cluster.end.position<-pl.cl+df.freq.cluster.true
    true.cluster.end.position<-true.cluster.end.position[!is.na(true.cluste
r.end.position)]
    true.cluster.end.position<-sort(true.cluster.end.position,decreasing =
FALSE)

    df.cluster.position<-cbind(true.cluster.start.position,true.cluster.end
.position)


    df.cluster.position<-as.data.frame(df.cluster.position)


    n=length(df.cluster.position$true.cluster.start.position)

    start.time<-rep(NA,n)

    for(i in 1:n){
      start.time[i]<-df.cluster.time$time.cluster[df.cluster.position$true.
cluster.start.position[i]]
    }

    df.cluster<-as.data.frame(cbind(df.cluster.position$true.cluster.start.
position,df.cluster.position$true.cluster.end.position,start.time))
```

```
    #bind end.time
    end.time<-rep(NA,n)

    for(i in 1:n){
      end.time[i]<-df.cluster.time$time.cluster[df.cluster.position$true.cl
uster.end.position[i]]
    }

    df.cluster<-cbind(df.cluster,end.time)


    #bind Lowfreq.cluster

    Lowfreq.cluster<-rep(NA,n)

    for(i in 1:n){
      Lowfreq.cluster[i]<-min(df.cluster.time$Lowfreq[df.cluster.position$t
rue.cluster.start.position[i]],df.cluster.time$Lowfreq[df.cluster.position$
true.cluster.end.position[i]])
    }


    df.cluster<-cbind(df.cluster,Lowfreq.cluster)

    #bind Highfreq.cluster
    Highfreq<-as.numeric(as.vector(df.cluster.time$Highfreq))

    Highfreq.cluster<-rep(NA,n)

    for(i in 1:n){
      Highfreq.cluster[i]<-max(Highfreq[df.cluster.position$true.cluster.st
art.position[i]],Highfreq[df.cluster.position$true.cluster.end.position[i]]
)
    }

    df.cluster<-cbind(df.cluster,Highfreq.cluster)

    #bind height.cluster

    height.cluster<-df.cluster$Highfreq.cluster- df.cluster$Lowfreq.cluster

    df.cluster<-cbind(df.cluster,height.cluster)

    #bindd length.cluster

    length.cluster<-df.cluster$end.time- df.cluster$start.time

    df.cluster<-cbind(df.cluster,length.cluster)



    suppressMessages(df.cluster<-rename(df.cluster,c("V1"="start","V2"="end
")))
```

```
    ##bind df.cluster.time and df.cl.time.freq -> df.end
    cluster.name<-seq(from=1,by=1,to=(length(df.cluster.time$time.cluster))
)

    df.end<-cbind(cluster.name,
                  "start.time"=df.cluster.time$time.cluster,
                  "end.time"=df.cluster.time$time.cluster,
                  "low.freq"=df.cluster.time$Lowfreq,
                  "high.freq"=as.numeric(as.vector(df.cluster.time$Highfreq
)),
                  "height.cluster"=(df.cluster.time$size.cluster-1)*freq.di
f,
                  "length.cluster"=rep(0,(length(df.cluster.time$time.clust
er))))

    df.end<-as.data.frame(df.end)

    end<-df.cluster$end
    start<-df.cluster$start
    df.cluster$end<-NULL

    n<-length(df.end$cluster.name)
    n<-ifelse(n==0,1,n)

    for(i in 1:n){
      ifelse(is.element(df.cluster$start,i),df.end[i,]<-df.cluster[which(df
.cluster$start==i),],0)
    }

    df.end$cluster.name<-seq(from=1,to=n,by=1)


    n=length(end)

    seq.to.delet<-c(NA)

    for(i in 1:n){
      seq.to.delet[i]<-ifelse(is.element(end[i],start),NA,end[i])
    }
    seq.to.delet<-seq.to.delet[!is.na(seq.to.delet)]

    df.end<-df.end[-seq.to.delet, ]
    df.end.1<-df.end
    df.end.1[1,]<-rep(1,7)
    ifelse(is.na(df.end[1,1]),df.end<-df.end.1,1)



    #build cluster up to max.cluster.length
    y<-0
    repeat{
```

```
df.end.1[1,]<-rep(1,7)
ifelse(is.na(df.end[1,1]),df.end<-df.end.1,1)
pl.cl<-seq(from=1,to=length(df.end$cluster.name),by=1)

n=length(pl.cl)

#lowfreq1

Lowfrq1<-c(NA)

for(i in pl.cl){
  Lowfrq1[i]<-df.end$low.freq[i]
}
Lowfrq1<-Lowfrq1[!is.na(Lowfrq1)]

#Highfreq1+2

Highfreq<-df.end$high.freq

#Highfreq1
Highfrq1<-c(NA)

for(i in pl.cl){
  Highfrq1[i]<-Highfreq[i]
}
Highfrq1<-Highfrq1[!is.na(Highfrq1)]

df.freq.cluster.true<-data.frame(matrix(ncol=5,nrow=length(pl.cl)))
#str(df.freq.cluster.true)


for(x in 1:5){


  #lowfreq2
  Lowfrq2<-c(NA)

  for(i in pl.cl){
    Lowfrq2[i]<-df.end$low.freq[i+x]
  }
  Lowfrq2<-Lowfrq2[!is.na(Lowfrq2)]
  Lowfrq2



  #Highfreq2
  Highfrq2<-c(NA)

  for(i in pl.cl){
    Highfrq2[i]<-Highfreq[i+x]
  }
  Highfrq2<-Highfrq2[!is.na(Highfrq2)]
  Highfrq2
```

```
        n<-length(Lowfrq1)

        frq.cluster.true<-c(NA)

        for(i in 1:n){
          frq.cluster.true[i]<-
            ifelse(
              (Lowfrq1[i]==Lowfrq2[i]||
                 Lowfrq1[i]==Highfrq2[i]||
                 Highfrq1[i]==Lowfrq2[i]||
                 Highfrq1[i]==Highfrq2[i]||
                 Highfrq1[i]>=Highfrq2[i]&&
                 Lowfrq1[i]<=Highfrq2[i]||
                 Highfrq1[i]<=Highfrq2[i]&&
                 Highfrq1[i]>=Lowfrq2[i])
              ,1,NA)
        }

        for(i in 1:n){
          frq.cluster.true[i]<-
            ifelse(frq.cluster.true[i]==1&&
                     round(df.end$end.time[i]+time.dif,digits = 3)==
                     round(df.end$end.time[i+x],digits = 3),1,NA)
        }


        df.freq.cluster.true[,x]<-frq.cluster.true
      }

      true.cluster.start.position<-pl.cl*df.freq.cluster.true
      true.cluster.start.position<-true.cluster.start.position[!is.na(true.
cluster.start.position)]
      true.cluster.start.position<-sort(true.cluster.start.position,decreas
ing = FALSE)

      for(i in 1:5){
        df.freq.cluster.true[,i]<-df.freq.cluster.true[i]*i
      }

      #true cluster end position
      true.cluster.end.position<-pl.cl+df.freq.cluster.true
      true.cluster.end.position<-true.cluster.end.position[!is.na(true.clus
ter.end.position)]
      true.cluster.end.position<-sort(true.cluster.end.position,decreasing
= FALSE)

      df.cluster.position<-cbind(true.cluster.start.position,true.cluster.e
nd.position)


      #built df.cluster und bind start.time
      df.cluster.position<-as.data.frame(df.cluster.position)
```

```
      n=length(df.cluster.position$true.cluster.start.position)

      start.time<-rep(NA,n)

      for(i in 1:n){
        start.time[i]<-df.end$start.time[df.cluster.position$true.cluster.s
tart.position[i]]
      }

      df.cluster<-as.data.frame(cbind(df.cluster.position$true.cluster.star
t.position,df.cluster.position$true.cluster.end.position,start.time))

      #bind end.time
      end.time<-rep(NA,n)

      for(i in 1:n){
        end.time[i]<-df.end$end.time[df.cluster.position$true.cluster.end.p
osition[i]]
      }


      df.cluster<-cbind(df.cluster,end.time)

      #bind Lowfreq.cluster
      Lowfreq.cluster<-rep(NA,n)

      for(i in 1:n){
        Lowfreq.cluster[i]<-min(df.end$low.freq[df.cluster.position$true.cl
uster.start.position[i]],df.end$low.freq[df.cluster.position$true.cluster.e
nd.position[i]])
      }

      df.cluster<-cbind(df.cluster,Lowfreq.cluster)

      #bind Highfreq.cluster
      Highfreq<-as.numeric(as.vector(df.end$high.freq))

      Highfreq.cluster<-rep(NA,n)

      for(i in 1:n){
        Highfreq.cluster[i]<-max(Highfreq[df.cluster.position$true.cluster.
start.position[i]],Highfreq[df.cluster.position$true.cluster.end.position[i
]])
      }

      df.cluster<-cbind(df.cluster,Highfreq.cluster)


      #bind height.cluster
      height.cluster<-df.cluster$Highfreq.cluster- df.cluster$Lowfreq.clust
er

      df.cluster<-cbind(df.cluster,height.cluster)
```

```
      #bind length.cluster
      length.cluster<-df.cluster$end.time- df.cluster$start.time


      df.cluster<-cbind(df.cluster,length.cluster)


      suppressMessages(df.cluster<-rename(df.cluster,c("V1"="start","V2"="e
nd")))


      ##bind df.end and df.cl.time.freq -> df.end

      df.end.1[1,]<-rep(1,7)
      ifelse(is.na(df.end[1,1]),df.end<-df.end.1,1)


      end<-df.cluster$end
      end<-ifelse(is.na(end),0,end)

      start<-df.cluster$start
      start<-ifelse(is.na(start),0,start)

      df.cluster$end<-NULL

      n<-length(df.end$cluster.name)

      for(i in 1:n){
        ifelse(is.element(df.cluster$start,i),df.end[i,]<-df.cluster[which(
df.cluster$start==i),],0)
      }


      df.end$cluster.name<-seq(from=1,to=n,by=1)


      n=length(end)

      seq.to.delet<-c(NA)

      for(i in 1:n){
        seq.to.delet[i]<-ifelse(is.element(end[i],start),NA,end[i])
      }
      seq.to.delet<-seq.to.delet[!is.na(seq.to.delet)]

      n<-length(seq.to.delet)

      for(i in 1:n){
        seq.to.delet[i]<-ifelse(length(seq.to.delet)==0,1,seq.to.delet[i])
      }

      ifelse(seq.to.delet==0,0,df.end<-df.end[-seq.to.delet, ])
```

```
      y<-y+1
      if(y==cluster.times-1){
        break
      }

    }
    #####


    #Filter out clusters higher than max.cluster.height####
    df.end<-df.end[ ! (df.end$height.cluster) > (max.cluster.height*freq.di
f), ]
    #####


    #Filter out clusers lower than min.cluster.height####
    df.end<-df.end[ ! (df.end$height.cluster) < ((min.cluster.height-1)*fre
q.dif), ]
    #####

  }
    #df.end replace if empty####
    ifelse(length(df.end$cluster.name)==0,df.end[1,]<-rep(0,7),0)

    df.end.append<-df.end
    df.end.append$start.time<-df.end$start.time+(section.length.min*s*60)
    df.end.append$end.time<-df.end$end.time+(section.length.min*s*60)
    df.end.comp<-rbind(df.end.comp,df.end.append)
    #####




    #call seq detector#################################
    df.call.seq<-data.frame(matrix(ncol = 6,nrow = 1))
    colnames(df.call.seq)<-c("seq.start","seq.end","start.time","end.time",
"freq.seq","seq.length")

    for(n in 2:33){

      cluster.nr<-seq(from=1,to=length(df.end$cluster.nam),by=1)


      #low freq lower than frequency window
      which.low.freq<-which(df.end$low.freq<=n*freq.dif+1    )

      if.one<-ifelse(is.element(cluster.nr,which.low.freq),cluster.nr,0)


      #high freq higher than frequency window
      which.high.freq<-which(df.end$high.freq>=n*freq.dif+1 )

      if.two<-ifelse(is.element(cluster.nr,which.high.freq),cluster.nr,0)
```

```
#both
if.both<-ifelse(if.one==if.two,cluster.nr,0)


if.both<- if.both[ if.both !=0 ]

#start.time
if.both.start.time<-c(NA)

for( i in 1:length(if.both)){
  if.both.start.time[i]<-df.end$start.time[if.both[i]]
}

#difference of start.time
delta.time<-diff(if.both.start.time)


#threshold fix delet all delta.time over or under threshold

delta.time.corr<-c(NA)

for(i in 1:length(delta.time)){
  delta.time.corr[i]<-ifelse(delta.time[i]>threshold.fix.high||
                               delta.time[i]<threshold.fix.low,
                             0,delta.time[i])
}


delta.time.corr.values<- delta.time.corr[ delta.time.corr !=0 ]

# median
m.corr<-median(delta.time.corr.values)

ind.ok<-c(NA)

for(i in 1:length(delta.time.corr)){
  ind.ok[i]<-ifelse(delta.time.corr[i]>m.corr*median.threshold.high||
                      delta.time.corr[i]<m.corr*median.threshold.low,
                    0,if.both[i])
}


ind.ok.values<- ind.ok[ ind.ok !=0 ]

which.ind.ok<-which(ind.ok!=0)

diff.which.ind.ok<-diff(which.ind.ok)

seq.true.start<-c(NA)

for(i in 1:length(ind.ok)){
  seq.true.start[i]<-ifelse(ind.ok[i]>0&&ind.ok[i-1]>0,seq.true.start
```

```
[i-1],ind.ok[i])

        seq.true.start[i]<-ifelse(ind.ok[i]==0&&ind.ok[i-1]>0,seq.true.star
t[i-1],seq.true.start[i])

        seq.true.start[i]<-ifelse(ind.ok[i]==0&&ind.ok[i-1]==0,0,seq.true.s
tart[i])

        seq.true.start[i]<-ifelse(is.na(seq.true.start[i]),ind.ok[i],seq.tr
ue.start[i])
      }


      table.start<-data.frame(table(seq.true.start))
      seq.start<-as.numeric(as.vector(table.start$seq.true.start))
      seq.start<-seq.start[ seq.start !=0 ]


      #seq.end
      test<-which(is.element(if.both,seq.start)==TRUE)

      table.start.freq<-as.numeric(as.vector(table.start$Freq))

      ifelse(length(seq.start)==length(table.start.freq),0,table.start.freq
<-table.start.freq[-1])

      table.start.freq<-table.start.freq-1

      seq.end.pos<-test+table.start.freq

      seq.end<-if.both[seq.end.pos]

      cbind(seq.start,seq.end)

      df.call.seq.n<-as.data.frame(cbind(seq.start,seq.end))

      start.time<-df.end$start.time[seq.start]
      start.time<-start.time+s*section.length.min*60

      end.time<-df.end$end.time[seq.end]
      end.time<-end.time+s*section.length.min*60

      seq.length<-table.start.freq

      freq.seq<-rep(n*freq.dif+1,length(start.time))

      df.call.seq.n<-cbind(df.call.seq.n,start.time,end.time,freq.seq,seq.l
ength)

      df.call.seq<-rbind(df.call.seq,df.call.seq.n)
    }

    df.call.seq<-df.call.seq[-c(1), ]
```

```
    df.call.seq<-df.call.seq[!(df.call.seq$seq.length<min.seq.length),]
    #####


    #df.call.seq replace if empty####
    ifelse(length(df.call.seq$seq.start)==0,df.call.seq[1,]<-rep(0,6),0)
    df.call.seq.comp<-rbind(df.call.seq.comp,df.call.seq)

  }
  #####
  df.call.seq.comp<-df.call.seq.comp[-1,]
  df.call.seq.comp<-df.call.seq.comp[-(which(df.call.seq.comp$seq.start==0)
),]
  df.call.seq.comp<-df.call.seq.comp[order(df.call.seq.comp$start.time),]

  df.end.comp<-df.end.comp[-1,]







  #call.seq.new####
  seq.1.0.low<-NA
  seq.1.0.high<-NA
  for(c in 1:(wav$samples/wav$sample.rate)){
    seq.1.0.high[c]<-max(df.call.seq.comp$freq.seq[which(df.call.seq.comp$s
tart.time<=c&df.call.seq.comp$end.time>=c)])

    seq.1.0.low[c]<-min(df.call.seq.comp$freq.seq[which(df.call.seq.comp$st
art.time<=c&df.call.seq.comp$end.time>=c)])
  }

  seq.1.0.low[is.infinite(seq.1.0.low)]<-0
  seq.1.0.high[is.infinite(seq.1.0.high)]<-0


  seq.1.0.low.seq<-0
  for(d in 1:length(seq.1.0.low)){
    seq.1.0.low.seq[d]<-ifelse(seq.1.0.low[d]==0,0,1)
  }
  for(d in 2:length(seq.1.0.low)){
    seq.1.0.low.seq[d]<-ifelse(seq.1.0.low.seq[d]==1&
                                 seq.1.0.low.seq[d-1]==0,max(seq.1.0.low.se
q)+1,seq.1.0.low.seq[d])
  }
  for(d in 2:length(seq.1.0.low)){
    seq.1.0.low.seq[d]<-ifelse(seq.1.0.low.seq[d]==1,seq.1.0.low.seq[d-1],s
eq.1.0.low.seq[d])
  }
  for(d in 2:length(seq.1.0.low)){
    seq.1.0.low.seq[d]<-ifelse(seq.1.0.low.seq[d]==0,0,seq.1.0.low.seq[d]-1
)
  }
```

```
  seq.new.start<-NA
  seq.new.end<-NA
  seq.new.low.freq<-NA
  seq.new.high.freq<-NA

  for(e in 1:max(seq.1.0.low.seq)){
  seq.new.start[e]<-min(which(seq.1.0.low.seq==e))
  seq.new.end[e]<-max(which(seq.1.0.low.seq==e))
  seq.new.low.freq[e]<-min(seq.1.0.low[which(seq.1.0.low.seq==e)])
  seq.new.high.freq[e]<-max(seq.1.0.high[which(seq.1.0.low.seq==e)])
  }

  ifelse(seq.new.start==Inf,seq.new.start<-2,"")
  ifelse(seq.new.end==-Inf,seq.new.end<--2,"")
  ifelse(seq.new.low.freq==Inf,seq.new.low.freq<-freq.dif,"")
  ifelse(seq.new.high.freq==-Inf,seq.new.high.freq<--freq.dif,"")
  #####


  #create output####
  dir.create(paste(wd,"/output",sep=""))
  setwd(paste(wd,"/output",sep=""))

  #export calls.detected
  if(calls.detected){
    length<-length(df.end.comp$cluster.name)

    Header<-c("Selection","View","Channel","Begin Time (s)","End Time (s)",
"Low Freq (Hz)","High Freq (Hz)","Annotation")
    Selection<- seq(from=1,to=length,by=1)
    View<- rep("Spectrogram 1",length = length)
    Channel<-rep(1,length = length)
    BeginTime<-df.end.comp$start.time
    EndTime<-df.end.comp$end.time
    Lowfreq<-df.end.comp$low.freq*1000
    Highfreq<-df.end.comp$high.freq*1000
    DeltaTime<- rep("",length = length)
    Deltafreq<- rep("",length = length)
    AvgPowerDensity<- rep("",length = length)
    Annotation<- df.end.comp$cluster.name

    input_raven<-data.frame(Selection,View,Channel,BeginTime,EndTime,Lowfre
q,Highfreq,Annotation)

    names(input_raven)<-Header

    write.table(input_raven,paste(list.f[z],"_calls.detected.txt",sep=""),r
ow.names = FALSE,dec=".",sep="\t",quote=FALSE)
  }

  #export call.seq.lines
```

```
  if(lines.call.seq){

    if(is_empty (df.call.seq.comp$seq.start)){df.call.seq.comp[1,]<-c(0,0,0
,0,0,0,0)}

    length<-length(df.call.seq.comp$start.time)

    Header<-c("Selection","View","Channel","Begin Time (s)","End Time (s)",
"Low Freq (Hz)","High Freq (Hz)","Annotation")
    Selection<- seq(from=1,to=length,by=1)
    View<- rep("Spectrogram 1",length = length)
    Channel<-rep(1,length = length)
    BeginTime<-df.call.seq.comp$start.time
    EndTime<-df.call.seq.comp$end.time
    Lowfreq<-df.call.seq.comp$freq.seq*1000
    Highfreq<-df.call.seq.comp$freq.seq*1000
    DeltaTime<- rep("",length = length)
    Deltafreq<- rep("",length = length)
    AvgPowerDensity<- rep("",length = length)
    Annotation<- df.call.seq.comp$seq.length

    input_raven<-data.frame(Selection,View,Channel,BeginTime,EndTime,Lowfre
q,Highfreq,Annotation)

    names(input_raven)<-Header

    write.table(input_raven,paste(list.f[z],"_call.seq.lines.txt",sep=""),r
ow.names = FALSE,dec=".",sep="\t",quote=FALSE)
  }

  #export call.seq
  if(call.seq){

    length<-length(seq.new.end)

    Header<-c("Selection","View","Channel","Begin Time (s)","End Time (s)",
"Low Freq (Hz)","High Freq (Hz)","Annotation")
    Selection<- seq(from=1,to=length,by=1)
    View<- rep("Spectrogram 1",length = length)
    Channel<-rep(1,length = length)
    BeginTime<-seq.new.start-2
    EndTime<-seq.new.end+2
    Lowfreq<-seq.new.low.freq*1000-freq.dif*1000
    Highfreq<-seq.new.high.freq*1000+freq.dif*1000
    DeltaTime<- rep("",length = length)
    Deltafreq<- rep("",length = length)
    AvgPowerDensity<- rep("",length = length)
    Annotation<- ifelse(length==1&seq.new.end==-2,"",rep("ALOB",length = le
ngth))

    input_raven<-data.frame(Selection,View,Channel,BeginTime,EndTime,Lowfre
q,Highfreq,Annotation)

    names(input_raven)<-Header
```

```
    write.table(input_raven,paste(list.f[z],"_call.seq.txt",sep=""),row.nam
es = FALSE,dec=".",sep="\t",quote=FALSE)
  }

}


t2<-Sys.time()
print(t2-t1)
delta.t<-t2-t1
setwd(wd)

}
```

Anhang VII

R-Skript zur Funktion “subset.output”

```
subset.output<-function(){
  wd<-getwd()
  setwd(paste(wd,"/output",sep = ""))
  list.of.files<-list.files(pattern = ".txt")

  for(i in 1:length(list.of.files)){
    setwd(paste(wd,"/output",sep = ""))
    data<-read.delim(list.of.files[i])
    data.subset<-subset(data,!data$Annotation=="")
    Header<-c("Selection","View","Channel","Begin Time (s)","End Time (s)",
"Low freq (Hz)","High freq (Hz)","Annotation")
    colnames(data.subset)<-Header
    write.table(data.subset,paste(list.of.files[i],sep=""),row.names = FALS
E,sep="\t",quote = FALSE)
  }
  setwd(wd)
}
```

Anhang VIII

R-Skript zur Funktion “join.output”

```
setwd("C:/Users/pmar/Dropbox/Data Manual")

join.output<-function(namecsv="joined.output"
){
  if(!require(data.table)){
    install.packages("data.table")}
  library(data.table)

  wd<-getwd()
  setwd(paste(wd,"/output",sep=""))



  list_of_files <- list.files(recursive = FALSE, pattern = ".txt")
  Geraet<-strsplit(list_of_files[1],"_")[[1]] [1]

  # Read all the files and create a FileName column to store filenames
  DT <- rbindlist(sapply(list_of_files, fread, simplify = FALSE),
                  use.names = TRUE, idcol = "FileName",fill = TRUE)
  DT<-cbind(Geraet,DT)
  DT[is.na(DT)]<-0
  DT$Annotation[DT$Annotation==""]<-0


  name<-DT$FileName


  date<-as.numeric(sapply(strsplit(name,"_"),"[[",2))
  time<-(sapply(strsplit(name,"_"),"[[",3))
  time<-as.numeric(gsub(".wav", "", time))


  DT<-cbind(DT,time)


  #noch bearbeiten
  rawDateChar = as.character(date)
  dateChar = paste0(stringr::str_sub(start=1,end=4,rawDateChar),"-",stringr
::str_sub(start=5,end=6,rawDateChar),"-",stringr::str_sub(start=7,end=8,raw
DateChar))
  date1 = as.Date(dateChar)


  DT<-cbind(DT,date)
  DT<-cbind(DT,date1)

  length<-length(DT$Selection)
  Geraet<-rep(Geraet,length=length)
```

```
  time.h<-floor(time/10000)/24
  time.m<-(time/10000-floor(time/10000))*100/60/24
  time1<-time.h+time.m

  real.time.begin<-(DT$`Begin Time (s)`/60/60/24)+time1
  real.time.end<-(DT$`End Time (s)`/60/60/24)+time1

  DT<-cbind(DT,real.time.begin)
  DT<-cbind(DT,real.time.end)

  DT$`Delta Time (s)`<-DT$`End Time (s)`-DT$`Begin Time (s)`

  write.table(DT,row.names = TRUE, col.names=NA,file = paste(namecsv,".csv"
,sep=""),sep=";",dec=".")
  setwd(paste(wd))
}
```

Anhang IX

R-Skript zur Funktion “create.soundfile”

```
create.soundfiles<-function(){
  wd<-getwd()
  setwd(paste(wd,"/output",sep=""))
  list.t<-list.files(pattern=".txt")

  dir.create(paste(getwd(),"/soundfiles",sep=""))

  setwd(paste(wd))
  list.f<-list.files(pattern=".wav")

  for(z in 1:length(list.f)){
    wav.entire<-readWave(paste(list.f[z]))

    setwd(paste(wd,"/output",sep=""))
    txt.table<-read.table(paste(list.t[z]),sep="\t",header=TRUE)

    setwd(paste(wd,"/output/soundfiles",sep=""))

    if(length(txt.table$End.Time..s.)>0){
      for(f in 1:length(txt.table$Begin.Time..s.)){
        writeWave(wav.entire[
          ((txt.table$Begin.Time..s.[f])*8000):(
            (txt.table$End.Time..s.[f])*8000)
        ],paste(list.f[z],"_seq_",f,".mp3",sep=""),extensible = FALSE)
      }
    }
    setwd(paste(wd))
  }
  setwd(paste(wd))
}
```

Anhang X

R-Skript zu Berechnung der Koordinaten

```
library(smacpod)
setwd("C:/Users/patri/OneDrive - ZHAW/Studium/6. Semester/BA/Koordinaten be
rechnen")

# Koordinaten der Fixpunte
# Punkte<-read.delim("Punkte.csv",sep=";",header = TRUE)
p1<-c(2648926.801,1186253.901)
p2<-c(2648889.101,1186230.751)
Punkte<-as.data.frame(cbind(p1,p2))

# Distanz zwische den Fixpunkten und den gesuchten Punkten (P1 und P2)
# Data<-read.delim("Data.csv",sep=";",header = TRUE)
# Distantz zwischen Fixpunt 1 und P1,P2
dp1<-c(14.36,31.5)
# Distantz zwischen Fixpunt 2 und P1,P2
dp2<-c(34.96,18.97)
# Seite, auf welcher P1,P2 liegen (Blickrichtung von Fixpunkt 1 zu Fixpunkt
2)
side<-c("right","right")
# Namen der Punte
name<-c("M1","M2")
Data<-as.data.frame(cbind(dp1,dp2,side,name))




p1<-c(Punkte$P1[1],Punkte$P1[2])
p2<-c(Punkte$P2[1],Punkte$P2[2])

coordinates.list<-NA

for(x in 1:length(Data$dp1)){

  dp1<-Data$dp1[x]
  dp2<-Data$dp2[x]

  side<-Data$side[x]

  cir = as.data.frame( cbind(c(p1[1],p2[1]), c(p1[2],p2[2]),c(dp1,dp2)),col
names<-c("x","y","r"))
  colnames(cir)<-c("x","y","r")




  i<-1

  # dx and dy are the vertical and horizontal distances between the circle
centers.
```

```
  dx = cir$x[i+1] - cir$x[i]
  dy = cir$y[i+1] - cir$y[i]

  # Determine the straight-line distance between the centers.

  d = sqrt((dy^2) + (dx^2))


  # Check for solvability.

  #no solution. circles do not intersect.
  ifelse(d > (cir$r[i] + cir$r[i+1]),FALSE,TRUE)

  # no solution. one circle is contained in the other
  ifelse(d < (cir$r[i] - cir$r[i+1]),FALSE,TRUE)



  # 'point 2' is the point where the line through the circle intersection p
oints crosses the line between the circle centers.

  # Determine the distance from point 0 to point 2.
  a = ((cir$r[i]^2) - (cir$r[i+1]^2) + (d^2)) / (2.0 * d)

  #Determine the coordinates of point 2.
  xm = cir$x[i] + (dx * a/d)
  ym = cir$y[i] + (dy * a/d)

  #Determine the distance from point 2 to either of thev intersection point
s.
  h = sqrt((cir$r[i]^2) - (a^2))

  #Now determine the offsets of the intersection points from point 2.
  rx = -dy * (h/d)
  ry = dx * (h/d)

  #Determine the absolute intersection points.

  xi0 = as.numeric(xm + rx)
  xi1 = as.numeric(xm - rx)
  yi0 = as.numeric(ym + ry)
  yi1 = as.numeric(ym - ry)

  cor.int<-rbind(c(xi0,yi0),c(xi1,yi1))
  colnames(cor.int)<-c("x","y")
  row.names(cor.int)<-c("i1","i2")
  cor.int<-as.data.frame(cor.int)
  cor.int



  # Prufen welcher der beiden Punkte der richtige ist
  d1<-ifelse(p2[2]-p1[2]==0,1,0)
  d2<-ifelse(p2[2]-p1[2]>0,1,0)
```

```
  e<-ifelse(d1==1&&p2[1]-p1[1]>0,which(cor.int$y==min(cor.int$y)),NA)
  f<-ifelse(d2==1,which(cor.int$x==max(cor.int$x)),which(cor.int$x==min(cor
.int$x)))

  g<- ifelse(is.na(e),f,e)

  h<- ifelse(side=="right",g,ifelse(g==1,2,1))

  coordinates<-cor.int[h,]
  coordinates

  coordinates.list<-rbind(coordinates.list,coordinates)
}
coordinates.list<-coordinates.list[-1,]
View(coordinates.list)

coordinates.list<-as.data.frame(cbind(coordinates.list,Data$name))
write.table(coordinates.list,"coordinates.list.csv",sep=";")
```

Anhang XI

R-Skript zur Lokalisierung von Schallquellen inkl. Clusterbildung

```
setwd("C:/Users/pmar/OneDrive - ZHAW/Studium/6. Semester/BA/Lokalisierung/data27")


library(tuneR)
library(SoundFinder)
library(raster)
library(seewave)
library(sp)
library(rgdal)
library(geosphere)



#parameter
temp = 12
cut<-0.4
wl=30
from.ts=0.55
to.ts=2.05
freq.band<-50
cut.ts<-60
plot<-F
plot.d<-F
wl.ts<-30
channel<-2
treshold.from<--22
treshold.to<--18


from.call<-1
to.call<- 60 #length(call.table$Selection)


# wav.to.ts
wav<-call1.f

wav.to.ts<-function(wav,wl=70,from=1.2,to=1.6){

spec<-spectro(wav,wl=wl,plot = F,flim = c(from,to))
spec.t<-(spec$amp[1,]+spec$amp[2,])/2

ts.l<-as.ts(smooth.spline(spec.t))
ts<-ts.l$y

return(ts)
}
```

```
apd4<-median(call.table$Avg.Power.Density..dB.FS.[from.call:to.call])

# wav. einlesen
wav.Files <- lapply(Sys.glob("*.wav"), readWave,to=4*60*24000)
wav<-wav.Files[[1]]
cutw(wav,from = 1,to= seewave::duration(wav),output = "Wave",channel = chan
nel)


# txt einlesen
call.list <- lapply(Sys.glob("*.txt"), read.delim)
call.table <- call.list[[1]]


#time lag loop
sounds.1<-NA
for(i in from.call:to.call){

  #from to
  from.filter = (call.table$Low.Freq..Hz.[i]+call.table$High.Freq..Hz.[i])/
2-freq.band/2
  to.filter = (call.table$Low.Freq..Hz.[i]+call.table$High.Freq..Hz.[i])/2+
freq.band/2


  # cut filter to ts loop
  df.ts<-NA
  for(n in 1:length(wav.Files)){
    #cut
    call<- cutw(wav.Files[[n]], from=call.table$Begin.Time..s.[i]-(cut/3),
to=call.table$Begin.Time..s.[i]+(2*cut/3), output="Wave",channel=channel)

    #filter
    call.f<- bwfilter(wave = call, n=3, from = from.filter, to = to.filter,
output = "Wave")

    #to ts
    ts<-wav.to.ts(call.f,wl = wl,from = from.ts,to = to.ts)
    ts<-ts[cut.ts:length(ts)-cut.ts]

    df.ts<-rbind(df.ts,ts)
  }
  df.ts<-df.ts[-1,]

  #plot ts
  ifelse(plot==T,plot.ts(),0)


  #ccf????

  time.step<-seewave::duration(call)/length(ts)

  #vec.t loop
```

```
  vec.t<-NA
  for(y in 1:length(wav.Files)){

    ts.h.t<-NA
    for (t in seq(treshold.from,treshold.to)){
      ts.h<-which(df.ts[y,]>=t)
      ts.h<-ts.h[ts.h > 40]
      ts.min<-min(ts.h)
      ts.min<-ifelse(ts.min<100,NA,ts.min)
      ts.h.t<-c(ts.h.t,ts.min)
    }
    ts.h.t<-ts.h.t[-1]
    res.samp<-median(ts.h.t)


    res.t<-(res.samp+cut.ts)*time.step


    test<-res.t-(cut.ts*time.step)

    vec.t<-c(vec.t,test)
  }


  vec.t<-vec.t[-1]



  #create df sounds
  sounds.loop<-NA
  for (a in 1:length(wav.Files)){
    sounds.loop<-cbind(sounds.loop,vec.t[a])
  }
  sounds.1<-rbind(sounds.1,sounds.loop)

}
sounds.1<-sounds.1[-1,]
sounds.1<-sounds.1[,-1]

##
sounds<-NA
for(i in 1:length(sounds.1[,1])){
  s<-sounds.1[i,]-(min(sounds.1[i,1:length(sounds.1[1,])],na.rm = TRUE))

  sounds<-rbind(sounds,s)
}
sounds<-sounds[-1,]
sounds<-as.data.frame(cbind(temp,sounds))
sounds[sounds == Inf] <- NA


#colnames sounds
c.names<-"temp"
```

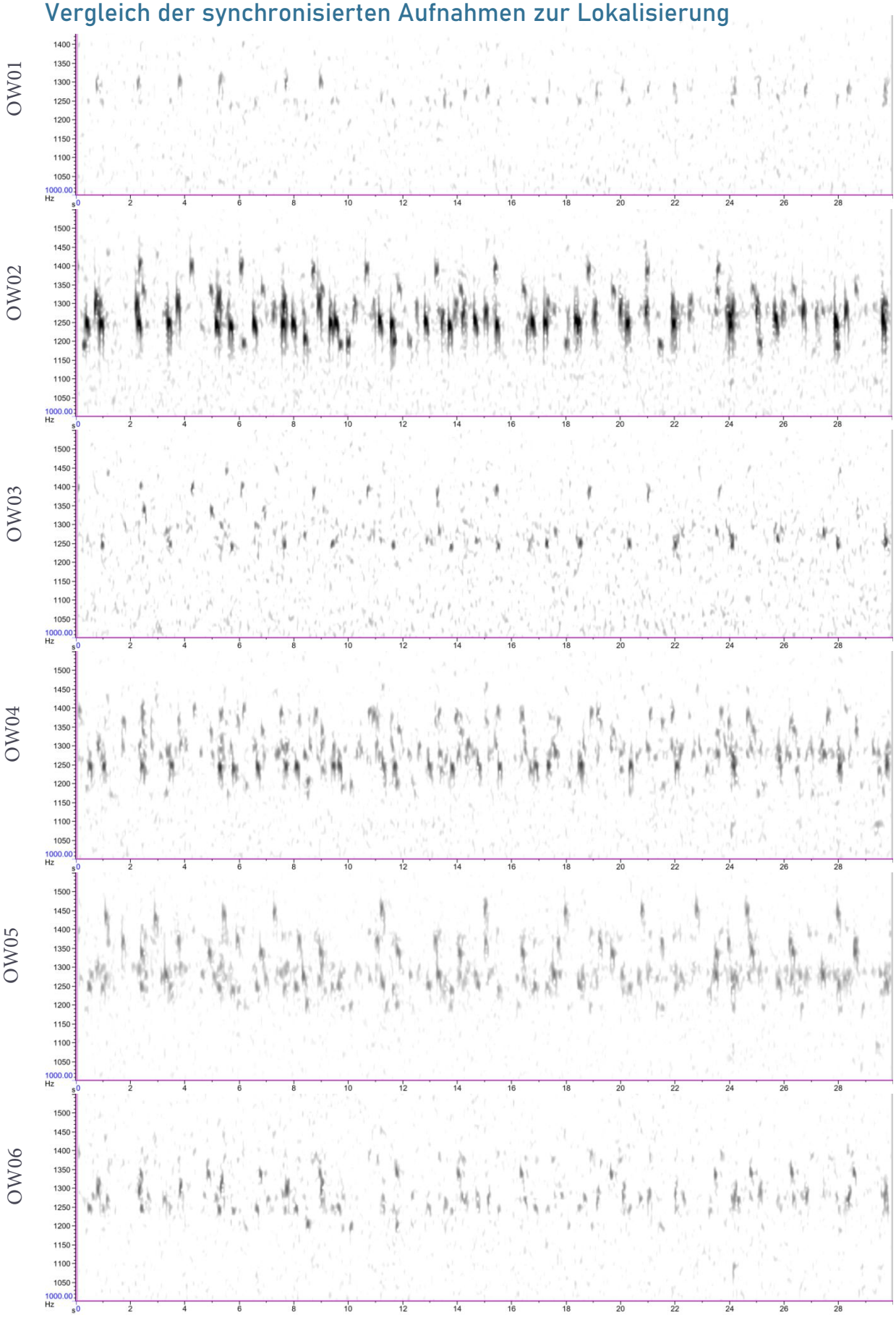
```
for(i in 1:length(wav.Files)){
  c.names<-c(c.names,paste("t",i,sep=""))
}

colnames(sounds)<-c.names
rownames(sounds)<-seq(1:length(sounds$temp))
sounds
sounds[which(rowSums(sounds)-temp==0),]<-rep(NA,length(sounds))
sounds[as.numeric(which(rowSums(is.na(sounds))>3)),]<-rep(NA,length(sounds)
)


#read df mics
mics<-read.delim("mics.csv",sep = ";",dec = ".")
mics

#localisation
sound.locations<-localize(mics,sounds)



#cluster analysis
x<-sound.locations$east[which(sound.locations$err.metres<1)]
y<-sound.locations$north[which(sound.locations$err.metres<1)]


xy<-as.data.frame(cbind(x,y))

mdist<-dist(xy)

# cluster all points using a hierarchical clustering approach
hc <- hclust(as.dist(mdist), method="complete")

# define the distance threshold, in this case 40 m
d=15

# define clusters based on a tree "height" cutoff "d" and add them to the S
pDataFrame
clust <- cutree(hc, h=d)

tab<-as.data.frame(table(clust))

t.f<-is.element(clust,which(tab[,2]>=5))

xy<-cbind(xy,clust)
t.f.n<-which(t.f==T)

xy.n<-xy[t.f.n,]

count.cluster<-length(table(xy.n$clust))

median.x<-NA
sd.x<-NA
median.y<-NA
```

```
sd.y<-NA
for(i in 1:length(unique(xy.n$clust))){
median.x.1<-median(xy.n$x[which(xy.n$clust==unique(xy.n$clust)[i])])
sd.x.1<-sd(xy.n$x[which(xy.n$clust==unique(xy.n$clust)[i])])
median.y.1<-median(xy.n$y[which(xy.n$clust==unique(xy.n$clust)[i])])
sd.y.1<-sd(xy.n$y[which(xy.n$clust==unique(xy.n$clust)[i])])
median.x<-c(median.x,median.x.1)
sd.x<-c(sd.x,sd.x.1)
median.y<-c(median.y,median.y.1)
sd.y<-c(sd.y,sd.y.1)
}

median.x<-median.x[-1]
sd.x<-sd.x[-1]
median.y<-median.y[-1]
sd.y<-sd.y[-1]

median.xy<-as.data.frame(cbind(median.x,median.y))
colnames(median.xy)<-c("x","y")

#statisitik####
alldist <- as.matrix( dist( rbind(median.xy,so) ) )
alldist[alldist == 0] <- 100

min.dist<-NA
for (i in 1:count.cluster){
  min.dist.l<-min( alldist[i,] )
  min.dist<-c(min.dist,min.dist.l)
}
min.dist<-min.dist[-1]

count.cluster
min.dist
sd.x
sd.y

#Visu
jpeg("Darstellung 2.jpeg", width = 1000, height = 691, res = 150 , pointsiz
e = 10)

map<-brick("Gisibach_1.tif")
par(mar=c(4,5,2,2))
plot(c(2691720,2691810),c(1230140,1230200),
     col="white",
     xlab=NA,ylab=NA,
     frame.plot=FALSE,
     axes=FALSE)
axis(1,pos=1230140,at=c(2691725,2691750,2691775,2691800),labels = c("2'691'
725","2'691'750","2'691'775","2'691'800"))
axis(2,pos=2691720,at=c(1230150,1230175,1230200),labels = c("1'230'150","1'
230'175","1'230'200"))


plotRGB(map,axes = FALSE,ext= c(2691720,2691810,1230140,1230200),maxpixels=
```

```
map@ncols*map@nrows,interpolate=T,add=TRUE)
rect(2691720,1230140,2691810,1230240,col = rgb(1,1,1,0.35),border = NA)

# grid
axis(1,pos = 1230140, at = seq(2691720,2691820,10) , tck = 1, lty = 2, col
= "grey", labels = NA)
axis(2,pos = 2691720, at = seq(1230150,1230200,10) , tck = 1, lty = 2, col
= "grey", labels = NA)

# calculated sounds
points(median.xy,col="blue",pch=16)
text(median.xy$x+1,median.xy$y-3,c(1,2,3,4))

# Vertical arrow
arrows(x0=median.xy$x, y0=median.xy$y-sd.y, x1=median.xy$x, y1=median.xy$y+
sd.y, code=3, angle=90, length=0.05, col="blue", lwd=1)
# Horizontal arrow
arrows(x0=median.xy$x-sd.x, y0=median.xy$y, x1=median.xy$x+sd.x, y1=median.
xy$y, code=3, angle=90, length=0.05, col="blue", lwd=1)



#mics
mics<-read.delim("mics.csv",sep=";")
points(mics$east,mics$north,col="red",pch = 16)
text(mics$east+2,mics$north+1,c(1,2,3))

#sound origin
so<-read.delim("so.csv",sep=";")
points(so[,1],so[,2],col="orange",pch=16)


#legend
rect(2691775,1230155,2691810,1230175,col = "white")
legend(2691775,1230175, legend=c("Mikrofone","Schallquelle","Mittelpunkt de
r Cluster"),
       col=c("red","orange","blue"), pch = 16,
       box.lwd = NA)
rect(2691775,1230155,2691810,1230175)


#scale bar
px<-c(2691780,2691790)
py<-c(1230159,1230159)
points(px,py,type = "l")
points(c(px[1],px[1]),c(py[1]-1,py[1]+1),type = "l")
points(c(px[2],px[2]),c(py[1]-1,py[1]+1),type = "l")
legend(px[1]-2,py[1]+1,legend="10m",bty="n")

north.arrow(2691802,1230159,len = 0.8,lab = "N",col="grey")
dev.off()
```

Anhang XII

Vergleich der synchronisierten Aufnahmen zur Lokalisierung



Anhang XIII
Anleitung zur automatischen Erkennung von Geburtshelferkröten

Vorbereitung
Installieren sie die Programme RStudio und RavenLite von folgenden Links:

- https://rstudio.com/products/rstudio/download/
- http://ravensoundsoftware.com/raven-pricing

Laden sie die R-Skripte „Functions.R“ und „Workflow.R“ sowie den Ordner „Soundfiles“ von folgendem Link herunter und speichern sie diese ab:

- https://www.dropbox.com/sh/hb7gl4l344gckuq/AABKRAVziOzT3CxAbBs6ZRsYa?dl=0

Der Ordner „soundfiles“ enthält folgende drei Soundfiles (Dauer je eine Stunde):

- OW03_20190518_190000.wav enthält Geburtshelferkröten-Rufe
- OW03_20190518_200000.wav enthält Störgeräusche (Kuhglocken)
- OW03_20190518_210000.wav enthält keine besonderen Geräusche

(Benennung: [Gerät]_JJJJMMTT_hhmmss.wav)

Funktionen definieren
Starten sie das Programm RStudio und öffnen sie den R-Skript „Functions.R“ mit File > Open File. Dieses Skript enthält folgende Funktionen:

- `find.alob`
- `subset.output`
- `join.output`
- `create.soundfiles`

Mit Ctrl+A markieren sie nun das gesamte Skript und führen dieses mit Ctrl+Enter aus. Im Bereich „Environment“ sind die Funktionen nun unter „Functions“ abgespeichert und können angewandt werden.

Daten analysieren
Öffnen sie Anschliessend den R-Skript „Workflow.R“ mit File > Open File.

Zeilen, die mit `#` beginnen sind Kommentare, die restlichen sind Codezeilen. Um eine Codezeile auszuführen, klicken sie auf die entsprechende Zeile, und drücken dann Ctrl+Enter.

Im ersten Schritt stellen sie das Arbeitsverzeichnis (Working Directory) ein. Dies soll dem Ordner entsprechen, in dem die Audiodateien abgespeichert sind. Passen sie dazu den Pfad in Linie 4 an und führen sie diese mit Ctrl+Enter aus und kontrollieren sie, ob im Bereich „Console“ eine Fehlermeldung erscheint. Der Pfad muss zwingend den Schrägstrich „/“ und nicht den umgekehrten Schrägstrich (Backslash) „\“ enthalten, sowie mit Anführungs- und Schlusszeichen versehen sein.

Beispiel:

```
setwd("C:/Users/pmar/Desktop/soundfiles")
```

Die Funktion `find.alob` (alob = Abkürzung *Alytes obstetricans)* zur Analyse der Audiodateien bietet drei Output-Optionen an. Es sind dies:

- ungefilterte Einzelsignale (`calls.detected`)
- Rufreihen pro Frequenzfenster (`lines.call.seq`)
- Rufreihe als Box (`call.seq`)

Output-Optionen, die in den Linien 7, 8 und 9 mit `TRUE` definiert sind, werden generiert, solche mit der Definition `FALSE` nicht. Der Schwellenwert, ab dem ein Signal erkannt wird, wird mit dem Parameter `threshold.factor` in Zeile 10 bestimmt.

Belassen sie die Optionen wie voreingestellt und führen sie die Code-Zeile 7 mit Ctrl+Enter aus.

Voreinstellungen:

```
#Analyse
find.alob(calls.detected = FALSE,
          lines.call.seq = FALSE,
          call.seq = TRUE,
          threshold.factor = 0.8)
```

Die Berechnung der Daten dauert einen Moment. Der Fortschritt wird im Bereich „Console“ angezeigt. Warnmeldungen können ignoriert werden. Sobald die Analyse abgeschlossen ist, erscheint im Ordner „soundfiles“ der Unterordner „output“. Dieser enthält die generierten Raven-Selection-Tables als .txt-Dateien. Diese können anschliessend in RavenLite geöffnet werden.

Output in RavenLite kontrollieren

Starten sie das Programm RavenLite und öffnen sie mit Ctrl+O das erste Soundfile aus dem Ordner „soundfiles“. Es öffnet sich ein Konfigurations-Dialog. Wählen sie „Page Sound“ und stellen sie im Feld „Paging“ folgende Werte ein und bestätigen sie mit OK.

- Page size: 180 seconds
- Page increment: 100 percent
- Step increment: 10 percent

Mit Ctrl+Shift+O laden sie anschliessend die entsprechende Raven-Selection-Table (gleiche Bezeichnung wie das Audiofile) aus dem Ordner „output“.

Öffnen sie mit Rechtsklick das Fenster „Configure View Axis“ und geben sie folgende Wert ein und bestätigen sie mit OK.

Time

- Position: 0 seconds
- Scale: 180 seconds/Line

Frequency

- Position: 1000 Hertz
- Scale: 500 Hertz/Line

Weitere Einstellungen nehmen sie anhand der gelben Markierungen in Abbildung 1 vor.

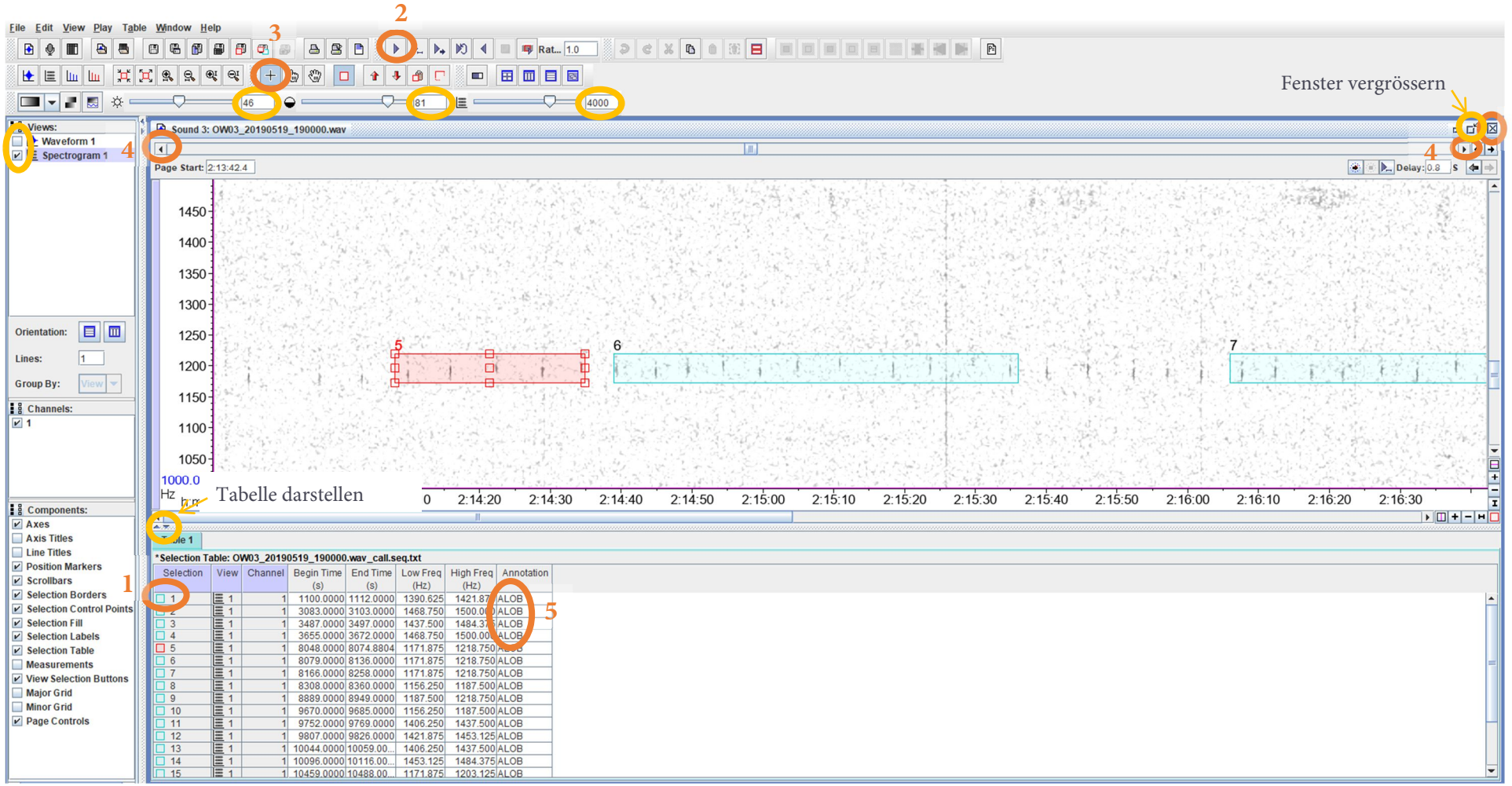


Abbildung 1: Einstellungen in RavenLite zur Darstellung von Geburtshelferkröten-Rufen. Dargestellt ist eine typische Rufreihe eines einzelnen Rufers.

Für später können Einstellungen mit View > Window Preset > Save as… mit einer entsprechenden Bezeichnung abgespeichert werden.

In dem sie bei 1 die erste Zeile auswählen, gelangen sie zur ersten Markierung. Beurteilen sie nun, ob es sich bei der Markierung tatsächlich um eine Rufreihe handelt. Mit 2 lässt sich die Aufnahme abspielen. Wenn eine Rufreihe gefunden wurde, kontrollieren sie ob die ganze Rufreihe erfasst ist. Passen sie die Box entsprechend den Rufen mit gehaltener Shift-Taste an, dazu muss 3 aktiviert sein. Mit 4 können sie auch die Abschnitte vor und nach dem aktuellen Abschnitt betrachten. Falls die Markierung keine Rufe enthält, löschen sie den Eintrag „ALOB“ bei 5. Mit den Pfeiltasten wählen sie anschliessend die nächste Markierung. Falls die Tabelle bei 1 nur eine Linie enthält und die Start- und Endzeit 0 beträgt, so konnten im ganzen Soundfile keine Rufe gefunden werden.

Schliessen sie das Fenster bei 6 und speichern sie Anpassungen ab. Anschliessend führen sie diesen Schritt mit allen Audiodateien durch. Im Konfigurations-Dialog können die abgespeicherten Einstellungen mit „Window Preset“ geladen werden („Configure View Axis“ muss erneut eingestellt werden).

Output aktualisieren und zusammenführen

Führen sie nun wieder im R-Skript „Workflow.R“ die Funktion `subset.output` in Linie 17 mit Ctrl+Enter aus. Diese entfernt alle Einträge aus allen Raven-Selection-Tabels, die keinen Eintrag in der Spalte „Annotaion“ enthalten (Einträge wie z. Bsp. „ALOB unsicher“ bleiben erhalten). Falls sie die ursprünglichen Resultate erhalten wollen, müssen diese vorgängig kopiert und abgespeichert werden.

Die Funktion `join.output` (Ctrl+Enter in Linie 20) fasst anschliessend die Raven-Selection-Tabels zu einer einzigen Tabelle zusammen und speichert diese als .csv-File im Ordner „output“ ab.

Abschliessend können die gefundenen Rufsequenzen mit der Funktion `create.soundfiles` (Ctrl+Enter in Linie 23) als entsprechend zugeschnittene Audiodaten abgespeichert werden. Diese finden sich im Unterordner „soundfiles“ im Ordner „output“.

Resultat

Die Datei „joined.output.csv“ im Ordner „output“ enthält nun alle wichtigen Informationen. Öffnen sie die Datei in Excel und formatieren sie die Spalten „real.time.begin“ und „real.time.end“ als Uhrzeit, um die Start- und Endzeit der Rufreihen korrekt anzuzeigen. Diese Datei dient dazu, weitere Analysen (zum Beispiel gesamte Rufdauer) durch zu führen. In Tabelle 1 sind die verschiedenen Spalten erklärt.

Tabelle 1: Bemerkung zu den Spalten der Datei „joined.output.csv“

Parameter	Bemerkung
Geraet	Gerätebezeichnung aus der Benennung des Soundfiles
FileName	Benennung des Soundfiles ([Gerät]_JJJJMMTT_hhmmss.wav)
Selection	Nummer der Markierung innerhalb eines Soundfiles
Channel	Aufnahmekanal des Aufnahmegeräts
Begin Time (s)	Startzeit der Markierung in Sekunden seit Beginn des Soundfiles
End Time (s)	Endzeit der Markierung in Sekunden seit Beginn des Soundfiles
Low freq (Hz)	Untergrenze der Markierung
High freq (Hz)	Obergrenze der Markierung
Annotation	Notiz, Voreinstellung: „ALOB“
time	Uhrzeit zu Beginn des Soundfiles [hhmmss]
date	Datum [JJJJMMTT]
date1	Datum [TT.MM.JJJJ]
real.time.begin	Uhrzeit beim Beginn der Markierung [s/24*60*60] mit Formatierung [hh:mm:ss]
real.time.end	Uhrzeit beim Ende der Markierung [s/24*60*60] mit Formatierung [hh:mm:ss]
Delta Time (s)	Dauer der Rufreihe

Anhang XIV

R-Skript Workflow

```
# Working Directory entsprechend dem Ordner einstellen, in dem die
# Audiodateien abgespeichert sind
setwd("C:/Users/pmar/Desktop/soundfiles")

# Analyse
find.alob(calls.detected = FALSE,
          lines.call.seq = FALSE,
          call.seq = TRUE,
          threshold.factor = 0.8)

# Mit Raven manuelle Uberprufung durchfuhren und bei
# falsch positiven in der Spalte Annotation den Eintrag "ALOB" loschen

# Entfernt falsch positive Eintrage aus dem Output und
# speichert im Order "Subset" neu ab
subset.output()

# Fasst die Resultate zusammen
join.output()

# Speichert die Resultate als Soundfiles
create.soundfiles()
```

Anhang XV

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit im Departement Life Sciences und Facility Management

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbständig und ohne fremde Hilfe verfasst hat. Der/die unterzeichnende Studierende erklärt, dass alle verwendeten Quellen (auch Internetseiten) im Text oder Anhang korrekt ausgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind. Bei Verfehlungen aller Art treten Paragraph 39 und Paragraph 40 der Rahmenprüfungsordnung für die Bachelor- und Masterstudiengänge an der Zürcher Hochschule für Angewandte Wissenschaften vom 29. Januar 2008 sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum: Unterschrift:

Felsberg, 13.01.2021

……