Karlsruher Institut für Technologie

Geräuscherkenner auf einem RaspberryPi 2

Bachelorarbeit von

Adrian Hein

an der Fakultät für Informatik
Institut für Anthropomatik und Robotik

Erstgutachter:	Prof. Alex Waibel
Zweitgutachter:	Dr. Sebastian Stüker

15. März 2016 – 14. Juli 2016

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

Karlsruhe, den 12.07.2016

. . . . . .

(Adrian Hein)

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 1
- 1.2 Überblick . . . 1

2 Grundlagen **3**
- 2.1 Mathematische Grundlagen . . . 3
  - 2.1.1 Spektralanalyse . . . 3
  - 2.1.2 Cepstralanalyse . . . 4
- 2.2 Methoden zur Merkmalsextraktion . . . 4
  - 2.2.1 FFT mit linearer Fensterung . . . 4
  - 2.2.2 Cepstrum mit linearer Fensterung ohne Liftering . . . 5
  - 2.2.3 Cepstrum mit linearer Fensterung mit Liftering . . . 6
- 2.3 Training der Modells . . . 6

3 Probleme und Lösungsansätze **9**
- 3.1 Begrenzte Rechenleistung . . . 9
  - 3.1.1 Ausnutzung der Quad-Core CPU . . . 9
- 3.2 Sammeln ausreichender Trainings- und Testdaten . . . 10

4 Experimente **11**
- 4.1 Allgemeine Experimente . . . 11
- 4.2 FFT mit lienarer Fensterung . . . 12
  - 4.2.1 Modell A: einfacher Pfeiffton . . . 12
  - 4.2.2 Modell B: Schließen einer Tür . . . 15
  - 4.2.3 Modell C: Ofentimer . . . 17
  - 4.2.4 Modell D: pfeifen von zwei Tönen . . . 19
  - 4.2.5 Modell E: Türklingel . . . 23
- 4.3 Cepstrum mit lienarer Fensterung ohne Liftering . . . 24
  - 4.3.1 Modell A: einfacher Pfeiffton . . . 24
  - 4.3.2 Modell B: Schließen einer Tür . . . 27
  - 4.3.3 Modell C: Ofentimer . . . 29
  - 4.3.4 Modell D: pfeifen von zwei Tönen . . . 29
  - 4.3.5 Modell E: Türklingel . . . 29
- 4.4 Cepstrum mit lienarer Fensterung mit Liftering . . . 30
  - 4.4.1 Modell A: einfacher Pfeiffton . . . 30
  - 4.4.2 Modell B: Schließen einer Tür . . . 32
  - 4.4.3 Modell C: Ofentimer . . . 33
  - 4.4.4 Modell D: pfeifen von zwei Tönen . . . 36

4.4.5 Modell E: Türklingel 39
4.5 Tests mit Hintergrundgeräuschen 41
4.5.1 FFT mit lienarer Fensterung 42
4.5.2 Cepstrum mit lienarer Fensterung ohne Liftering 44
4.5.3 Cepstrum mit lienarer Fensterung mit Liftering 46

5 Diskussion und Ausblick **47**
5.1 Diskussion 47
5.2 Ausblick 48

Literatur **49**

1 Einleitung

1.1 Motivation

Als Informatikstudent bin ich vom Prinzip eines Smart Home fasziniert. Ein großes Problem dabei ist allerdings die Eingliederung schon vorhandener Geräte ohne entsprechende Schnittstellen. Da viele Geräte wie z.B. Mikrowellen, Türklingel, Backöfen oder Küchetimer über akkustische Signale verfügen, um ein Ereignis oder eine Zustandsänderung anzuzeigen, bietet sich ein Geräuscherkenner an, um solche Geräte in die eigene Infrastruktur zu integrieren und auf entsprechende Ereignisse zu reagieren. Da Smart Homes nicht nur die Vernetzung von Geräten, sondern die allgemeine Erhöhung von Wohn- und Lebensqualität durch technische Systeme und Verfahren umfasst, sollte der Geräuscherkenner nicht nur technisch erzeugte, sondern alle Arten von Geräuschen erkennen. Dies könnte z.B. ein Klatschen oder Pfeifen, aber auch ein zuknallendes Fenster oder eine zuknallende Tür sein. Um möglichst flexibel reagieren zu können soll, nachdem ein Geräusch erkannt wurde, ein vom Benutzer hinterlegtes Python Script ausgeführt werden.

Damit der Geräuscherkenner möglichst flexibel eingesetzt werden kann, muss er sowohl klein als auch preisgünstig sein. Aufgrund dieser Anforderungen habe ich mich für einen RaspberryPi 2 Modell B (im folgenden als RaspberryPi bezeichnet) als Hardware entschieden. Da die Rechenleistung auf einem RaspberryPi doch eher beschränkt ist, mussten unterschiedliche Methoden zur Merkmalsextraktion und zur Erkennung eines Modells gegeneinander abgewogen werden. Kriterien für die Güte sind daher der Rechenaufwand, die Erkennungsrate sowie die Anzahl der False Positivs die erkannt werden.

Dieses System kann, obwohl es keine harten Echtzeitanforderungen erfüllt, durchaus als Echtzeiterkenner bezeichnet werden. Es baut zwar nicht auf einem echtzeitfähigen Betriebssystem auf, liefert im Normalbetrieb jedoch Ergebnisse in Echtzeit. Echtzeit bedeutet in diesem Fall, dass die Daten verarbeitet wurden und gegebenenfalls ein Ergebnis erkannt wird, bevor die nächsten Daten vorliegen. Da alle 23 *ms* neue Daten kommen, erfüllt das System im Normalbetrieb die Anforderung an ein Echtzeitsystem.

1.2 Überblick

Geräuscherkenner wurden bisher in der Forschung eher weniger betrachtet. Viel besser erforscht sind Spracherkenner. Deswegen gibt es heute schon gute Spracherkenner, auch für eher schwache Hardware wie den RaspberryPi. Diese sind allerdings auf Sprache optimiert und für andere Geräusche nur bedingt bis gar nicht nutzbar, da sie viel Vorwissen der Sprachproduktion nutzen.

Das einzig bekanntere Projekt, das einen Geräuscherkenner auch für den RaspberryPi mitentwickelt, ist das AudioSense Projekt der Firma Orelia [1]. Dies ist allerdings ein kommerzielles Projekt, dessen Quellcode nicht veröffentlicht wird. Leider veröffentlicht Orelia auch keine Zahlen, wie gut der Geräuscherkenner wirklich arbeitet, weswegen es sich nicht mit den hier vorgestellten Verfahren vergleichen lässt.

Es gibt allerdings einige Projekte, die Geräuscherkenner entwickelt haben, die nicht auf schwacher, sondern normaler Hardware laufen sollen. Ein Beispiel dafür wäre das CHIL Projekt, das unter anderem Geräusche erkennen und klassifizieren soll. Dafür wurden, wie in [7] beschrieben, verschiedene Merkmale und verschiedene Klassifizierer getestet. Da beim CHIL Projekt auch spektrale und cepstrale Merkmale verwendet wurden, lassen sich die Ergebnisse von diesem Projekt mit den hier erzielten Ergebnissen vergleichen.

Ein anderes Projekt nutzt “Temporal Independent Component Analysis (ICA)”, wie in [5] beschrieben. Dieses Projekt erzielt mit einer Erkennungsrate von 83,4 % schon recht gute Ergebnisse, und ist deutlich besser als das CHIL Projekt, das nur eine Erkennungsrate von 61,59 % erzielt hat.

Wie das in dieser Arbeit beschriebene Projekt im Vergleich mit den oben vorgestellten Projekten abschneidet, wird später in Experimenten gestestet werden.

2 Grundlagen

Audiodaten treten in der realen Welt, sowohl im Werte- als auch im Zeitbereich, als kontinuierliches Signal auf. Da in der digitalen Welt allerdings keine unendliche Genauigkeit gegeben sein kann, müssen Audiodaten als diskrete Werte abgespeichert werden. Die zeitliche Diskretisierung wird Sampling, die Diskretisierung des Wertebereichs Quantisierung genannt. Die entsprechenden Kennwerte sind einmal die Samplingrate (f_a), die in Hertz angegeben wird und aussagt, wieviel Datenpunkte pro Sekunde aufgenommen wurden, sowie die Bitrate, die angibt, wieviele Bit genutzt werden um einen einzelnen Datenpunkt zu speichern. Bei allen durchgeführten Experimenten wurde eine Samplingrate von 44100 Hz sowie eine Bitrate von 16 Bit genutzt.
Da eine Mustererkennung auf den oben beschriebenen Audiodaten schon allein aufgrund der reinen Datenmenge schwer zu realisieren ist, muss eine gute Methode gefunden werden, um möglichst aussagekräftige Merkmale zu erhalten. Diese sollten möglichst viel Information in möglichst wenig Daten enthalten.

2.1 Mathematische Grundlagen

2.1.1 Spektralanalyse

Eine Möglichkeit der Analyse ist, das Audiosignal aus dem Zeitbereich in den Frequenzbereich zu transformieren, um somit das Spektrum analysieren zu können. Dies funktioniert mathematisch über die sogenannte Fourier-Transformation, wobei $x(t)$ ein kontinueierliches Signal beschreibt.

$$F(x)(\omega) = \int_{-\infty}^{\infty} x(t) e^{-i\omega t}\,\mathrm{d}t$$

Um ein diskretes Signal zu transformieren, wird die Diskrete Fourier-Transformation F benutzt.

$$F(x[n]) = \sum_{n=-\infty}^{N-1} x[n] e^{-jk\frac{2\pi}{N}} = X(k), k = 0, 1, 2, \ldots, N-1$$

Wobei $x[n]$ das diskrete Signal beschreibt. Durch diese Transformation gehen keine Informationen verloren, was ganz einfach durch die inverse diskrete Fourier-Transformation F^{-1} gezeigt werden kann, die das transformierten Signal wieder zum originalen Signal transformiert.

$$F^{-1}(X(\omega)) = t_a \int_{-f_a}^{f_a} X(\omega) e^{i\omega n t_a} \, \mathrm{d}\omega = x[n]$$

Vergleiche dazu auch [4] Seite 208 ff. und Seite 216 ff.

2.1.2 Cepstralanalyse

Eine weiter Möglichkeit ist die Analyse des sogenannten Cepstrums, das als die inverse Fourier-Transformation des Logarithmus der Fourier-Transformation definiert ist.

$$F^{-1}(logF(x[n]))$$

Diese Transformation ergibt eine Funktion über die Quefrenz, und wird in der Sprachanalyse häufig eingesetzt.

2.2 Methoden zur Merkmalsextraktion

Es gibt verschiedene Verfahren zur Merkmalsextraktion. Ein mögliches Verfahren wäre zum Beispiel eine Diskrete Fourier-Transformation (DFT) anzuwenden.
Ein aus der Spracherkennung bekanntes Verfahren wären Mel Frequency Cepstral Coefficients (MFCC).
Ein weiteres Verfahren ist die "Temporal Independent Component Analysis (ICA)", wie in [5] beschrieben. Dieses Verfahren scheint gut geeignet zu sein, um Geräusche aus einer Küche zu erkennen, eignet sich allerdings weniger für den geplanten Geräuscherkenner, da mit über 4100 Geräuschen nur 21 verschiedene Events trainiert wurden. Da der Geräuscherkenner mit maximal 25 Trainingsaufnahmen pro Modell funktionieren soll, ist dies nicht praktikabel. Ein weiteres Problem wäre vermutlich die benötigte Rechenleistung der Merkmalsextraktion, die beim Trainieren zwar nicht so relevant ist, bei der Echtzeiterkennung allerdings schon.
Da auf dem Raspberry Pi nur eine begrenzte Rechenleistung verfügbar ist, und der Umfang dieser Arbeit nicht gesprengt werden sollte, wurden drei verschiedene Verfahren ausgewählt.

2.2.1 FFT mit linearer Fensterung

Eine recht einfache Methode zur Merkmalsextraktion ist die Berechnung der schnellen Fourier-Transformation (FFT von fast Fourier transform, eine schnelle Implementierung der Diskreten Fourier-Transformation) mit einer linearen Fensterung. Die schnelle Fourier-Transformation bildet ein zeitdiskretes, endliches Signal auf ein diskretes Frequenzspektrum ab. Da das Frequenzspektrum eines Signals, das mehrere Sekunden lang geht, wenig aussagekräftig ist, muss eine Fensterung vorgenommen werden. Eine Fensterung bezeichnet eine Zerlegung des Signals in viele kleine, je nach Art der Fensterung überlappender oder auch nicht überlappender Frames, auf denen dann jeweils eine FFT durchgeführt wird. In der Spracherkennung werden meist Frames mit einer Länge von 10-30 ms verwendet.

Somit wird nicht mehr das Frequenzspektrum des kompletten Signals, sondern eine Liste von Frequenzspektren zu verschiedenen Zeitpunkten betrachtet.
Bei der linearen Fensterung wird das Signal in gleich lange, sich nicht überlappender Frames eingeteilt. Diese Methode wurde gewählt, da sie sehr einfach zu implementieren ist, indem einfach immer eine bestimmte Anzahl X an Datenpunkten zusammengefasst werden, die FFT darauf angewendet wird und dann die nächsten X Datenpunkte zusammengefasst werden. Somit überlappen sich die Frames nicht und knüpfen direkt aneinander an.
Bei allen hier vorgestellten Methoden zur Merkmalsextraktion wurde eine Fensterlänge (n_f) von 1024 Datenpunkten gewählt. Durch die Samplingrate $r = 44100$ Hz lässt sich somit die zeitliche Länge (t) wie folgt bestimmen:

$$t = \frac{n_f}{r} = \frac{1024}{44100\,\text{Hz}} = 0,023219954648526\,\text{s} = 23,21\,\text{ms}$$

Eine FFT über 1024 Datenpunkten liefert einen Merkmale Vektor mit 1024 Werten zurück. Da sich in anfänglichen Tests zeigte, das dies jedoch viel zu viele Werte sind, um sie mit der begrenzten Rechenleistung des Raspberry Pi alle 23 ms zu vergleichen, müssen diese noch reduziert werden. Da die FFT für reelle Werte immer ein symmetrisches Ergebnis liefert, kann die Hälfte der Werte weggelassen werden ohne Informationen zu verlieren. Allerdings sind auch 512 Werte immer noch zu viel. Deswegen wurden immer 8 benachbarte Werte zu einem neuen Wert aufaddiert. Dadurch reduzieren sich die Werte auf $\frac{512}{8} = 64$ pro Frame. Dadurch geht zwar ein Teil der Informationen verloren, allerdsings zeigen die Experimente, dass der übriggebliebene Teil der Informationen ausreicht, um Muster erkennen zu können.

2.2.2 Cepstrum mit linearer Fensterung ohne Liftering

Eine weitere Methode zur Merkmalsextraktion ist die Berechnung des Cepstrums mit einer linearen Fensterung. Diese Methode wird vor allem in der Spracherkennung eingesetzt, dort allerdings mit überlappenden Fenstern, was sich hier nicht umsetzen ließ, da überlappende Fenster mit mehr Rechenaufwand verbunden wären, als die nicht überlappenden, die hier gewählt wurden. Diese Methode zur Merkmalsextraktion wird deswegen genutzt, da bei der Sprachproduktion von einem Quelle-Filter Modell ausgegangen wird. Ein Sprachsignal (f) besteht aus der Grundfrequenz (e), die durch die Stimmbänder erzeugt wird, und den Filtern (h) im Mund-/Rachenraum, die daraus die für Menschen hörbare Sprache modellieren.

$$f = e * h$$

Diese Faltung in der zeitabhängigen Funktion wird durch die Fourier-Transformation zu einer Multiplikation in der frequenzabhängigen Funktion.

$$F(f) = F(e) \times F(h)$$

Diese Multiplikation wird durch den Logarithmus zu einer Addition in der quefrenzabhängigen Funktion:

$$\log F(f) = \log F(e) + \log F(h)$$

$$F^{-1}(\log F(f)) = F^{-1}(\log F(e)) + F^{-1}(\log F(h))$$

Vergleiche dazu auch Kapitel 6.4 auf Seite 306 ff. in [4].

Dies bringt zwar keinen Vorteil bei der allgemeinen Geräuscherkennung, da hier nicht davon ausgegangen werden kann, dass jedes Geräusch durch ein Quellsignal mit Filtern erzeugt wird, allerdings lässt sich das Verfahren recht einfach implementieren. In den ersten Tests zeigte sich, dass dieses Verfahren eine recht hohe False Positivs Rate bei einem sehr einfachen Geräusch hatte. Die Toleranz, ob ein aufgenommener Frame zu einem Frame des Modells passt, war also recht hoch. Dies führt zwar bei einfachen Geräuschen zu den, wie eben erwähnten, hohen False Positivs Raten, könnte aber bei komplexeren Geräuschen von Vorteil sein. Da bei komplexeren Geräuschen mehr Frames pro Modell vorhanden sind, gleicht dies die hohe Toleranz aus. Obwohl für einen einzelnen Frame des Modells vielleicht mehr False Positivs erkannt werden, müssen, damit das Modell erkannt wird, alle Frames des Modells nacheinander erkannt werden. Dies könnte also bedeuten, dass dieses Verfahren gut für komplexere Geräusche geeignet ist. Ob dies wirklich der Fall wird durch die Experimente überprüft.

2.2.3 Cepstrum mit linearer Fensterung mit Liftering

In der Spracherkennung wird das sogenannte Liftering eingesetzt, um aus einem Cepstrum sowohl die Grundfrequenzen, als auch die Filterauswirkungen getrennt zu betrachten, da diese nach den oben beschriebenen Transformationen an unterschiedlichen Stellen im Merkmalsvektor liegen. Die Filterauswirkungen liegen dabei ca. im ersten Viertel des Vektors, die restlichen 3/4 beschreiben die Grundfrequenz. Da dies extrem einfach implementiert werden kann und so gut wie keine Rechenleistung verbraucht, wurde es zu Testzwecken mit implementiert. Ob es einen wirklichen Vorteil bringt, werden die Experimente zeigen.

2.3 Training der Modells

Um ein gutes Modell mit einer guten Erkennungsrate zu bekommen, genügt es nicht, die Merkmale aus den Aufnahmen zu extrahieren. Sie müssen auch irgendwie zusammengefasst werden. Dazu kann der User angeben, wieviele Zustände ein Modell haben soll. Danach werden die Differenzen zwischen allen benachbarten Fenstern gebildet. Die beiden Fenster mit der niedrigsten Differenz werden zusammengefasst. Dies wird so lange wiederholt, bis die gewünschte Anzahl Zustände erreicht wird. Danach wird für jeden Zustand des Modells die größte Differenz zwischen zwei Fenstern, die zu diesem Zustand zusammengefasst wurden, berechnet. Der Durchschnitt aus diesem Wert sowie der größten Differenz, bei dem überhaupt zwei Fenster zusammengefasst wurden, unabhängig davon, ob in diesem Zustand des Modells oder in einem anderen, wird als Schwellwert für diesen Zustand gewählt. Dieses Vorgehen verhindert, dass ein Zustand des Modells, der aus nur zwei sich sehr ähnlichen Fenstern der Aufnahme besteht, einen sehr kleinen Schwellwert

hat, der in der Echtzeiterkennung so gut wie nie erreicht wird. Beträgt während der Echtzeiterkennung die Differenz zwischen dem aktuell aufgenommenem Fenster und dem aktuellen Zustand des Modells weniger als der vorher berechnete Schwellwert, wird davon ausgegangen, den entsprechenden Zustand des Modells aufgenommen zu haben.
Um den Einfluss der eher unwichtigen Werte eines Zustands zu minimieren, kann der User festlegen, dass eine bestimmte Anzahl X der Werte pro Zustand weggelassen wird. Dies entspricht im Prinzip einer Filterbank bzw. mehreren Bandpassfiltern, da ein Wert im Zustand einem Frequenzband des Audiosignals entspricht. Es werden dann die kleinsten X Werte durch NaN (Not a Number) ersetzt und bei den Vergleichen nicht betrachtet. Dies bringt z.B. Vorteile, wenn ein Geräusch nur aus tiefen Frequenzen besteht, bei der Echtzeiterkennung allerdings hohe Frequenzen im Hintergrund auftreten. Da die hohen Frequenzteile im Zustand des Modells NaN sind, werden die hohen Frequenzen in der Aufnahme ignoriert und nur die tiefen Frequenzteile betrachtet.
Dies wird für alle Trainingsaufnahmen durchgeführt. Danach wird jede vorverarbeitete Aufnahme einmal als Ausgangsbasis zum Berechnen eines Modells gewählt. Da alle vorverarbeiteten Aufnahmen die gleiche Anzahl Zustände haben, wird der erste Zustand der ausgewählten Aufnahme mit jedem ersten Zustand der anderen Aufnahmen verglichen. Liegt die Differenz unter dem Schwellwert des Zustands der ausgewählten Aufnahme, so wird der Zustand vorgemerkt und am Schluss mit allen anderen ersten Zuständen, die vorgemerkt wurden, zum ersten Zustand des Modells zusammengefasst. Ausserdem wird die Anzahl der Zustände gespeichert, die zu diesem einen Zustand zusammengefasst wurden. Danach wiederholt sich das ganze Verfahren für den nächsten Zustand. Da durch dieses Verfahren das Modell stark von der ausgewählten Aufnahme abhängt, wird jede Aufnahme einmal ausgewählt und ein Modell daraus berechnet. Somit bekommt der User am Schluss genau so viele Modells wie Aufnahmen und kann sich unter diesen für eines entscheiden, das gespeichert wird.
Um entscheiden zu können, welches Modell vermutlich das beste ist, wird für jedes Modell ein Score berechnet. Hierfür werden alle Modells mit den Trainingsaufnahmen getestet. Der Score berechnet sich dann aus der Anzahl der erkannten Traingsaufnahmen (*matches*), dem Durchschnitt der Anzahl der Frames, die zu einem Frame des Modells zusammengefasst wurden (*influenced*), sowie dem Durchschnitt der Schwellwerte der einzelnen Frames (*threshold*).

$$\text{score} = \frac{matches \times influenced}{threshold}$$

Diese Berechnung wurde gewählt, da ein Modell um so besser ist, je mehr Aufnahmen erkannt wurden, je mehr Aufnahmen es beeinflusst haben und je geringer der Schwellwert ist.

3 Probleme und Lösungsansätze

3.1 Begrenzte Rechenleistung

Da der Geräuscherkenner auf einem RaspberryPi laufen sollte war von vornherin klar, dass die Rechenleistung ein Problem werden würde. Der RaspberryPi verfügt über eine 900MHz Quad-core CPU sowie über 1GB an Arbeitsspeicher.

Aufgrund dieser Einschränkung wurden nur die in Kapitel 2 beschriebenen Verfahren implementiert, obwohl es andere Verfahren gibt, die erfolgsversprechender wären. Allerdings sind diese mit einem größeren Rechenaufwand verbunden, und somit nicht praktikabel. Um die Prozessorleistung des RaspberryPi möglichst gut auszunutzen, wurden verschiedene Berechnungen parrallel ausgeführt.

3.1.1 Ausnutzung der Quad-Core CPU

Python bietet verschiedene Module um mehrere Threads oder Prozesse zu starten. Allerdings gibt es dabei einige Probleme.

3.1.1.1 threading Modul

Das threading Modul in Python kann genutzt werden, um mehrere Threads zu starten. Allerdings ist der Python Interpreter nicht thread safe. Dies bedeutet, dass zwar mehrer Threads gestartet werden können, der Python Interpreter allerdings immer nur einen Thread auf der CPU ausführt. Somit eignet sich dieses Modul nur für Anwendungen, bei der die einzelenen Threads auf eine Eingabe oder Daten aus einem Netzwerk oder ähnlichem warten. Solange sie warten, kann ein anderer Thread seine Anweisungen auf der CPU ausführen. Da bei dem hier beschriebenen Geräuscherkenner die Daten allerdings im Hauptspeicher vorliegen, hängt die benötigte Zeit für einen Funktionsaufruf nicht davon ab, wie schnell die Daten vorliegen, sondern wie schnell diese auf der CPU verarbeit werden. Somit bringt dieses Modul keinen Vorteil.

3.1.1.2 multiprocessing Modul

Das multiprocessing Modul arbeitet anders. Hier werden keine neuen Threads, sondern neue Python Prozesse mit jeweils einem eigenen Python Interpreter gestartet. Dadurch kann jeder Prozess auf einem eigenen CPU-Kern ausgeführt werden. Allerdings erschwert es die Kommunikation zwischen den Prozessen. Sie können z.B. nicht auf die gleichen Objekte zugreifen, da jeder Prozess seine eigenen Objekte hat. Ein weiteres Problem ist, dass die Reihenfolge, in der die Prozesse starten nicht festgelegt, sondern zufällig ist. Auch

der Endzeitpunkt kann variieren, wenn zum Beispiel das Betriebssystem einen wichtigen Task auf einem der Kerne ausführt, und der entsprechende Python Prozess auf diesem Kern solange stillgelegt wird.
Somit eignet sich dieses Modul nur für voneinander unabhängige Berechnungen. Diese liegen vor allem beim Berechnen eines Modells vor. Sowohl die Reduzierung auf einige wenige Frames, wie auch hinterher die Score Berechnung sind für jede Aufnahme beziehungsweise jedes Modell unabhängig voneinander aber rechenintensiv. Sie werden also parallel ausgeführt, und die Ergebnisse dann jeweils im Filesystem abgelegt. Die Funktionalität für das Ablegen im Filesystem muss sowieso gegeben sein, sonst müsste der User ja jedes Mal, wenn er den Geräuscherkenner neu startet, alle Modells neu berechnen. Da bei der Echtzeiterkennung die Reihenfolge der Frames wichtig ist und nicht vernachlässigt werden kann, bietet sich das multiprocessing Modul nicht an, um diese parallel berechnen zu können. Somit wurde für die Echtzeiterkennung nur sequentieller Code benutzt.

3.2 Sammeln ausreichender Trainings- und Testdaten

Eine weiteres Problem, das im Voraus schon bekannt war, war, wie genug Trainings- und Testdaten gesammelt und diese miteinander verglichen werden können. Um möglichst viele unterschiedliche Testsets zu erhalten, wurden zuerst verschiedene Geräusche, die später erkannt werden sollten, aufgenommen. Aus diesen konnte für jedes Geräusch ein Set in fixer Länge kombiniert werden, bei denen im Abstand von einer Minute die einzelnen Aufnahmen der Geräusche abgespielt wurden. Um später möglichst viele Testsets generisch zu bauen, musste darauf geachten werden, dass die Sets kompatibel zueinander sind, in keinen zwei Sets durfte also zum gleichen Zeitpunkt ein Geräusch aufkommen.

Da der Geräuscherkenner in der Praxis aber nicht nur in stillen Räumen funktionieren soll, mussten noch Hintergrundgeräusche eingefügt werden. Um ein möglichst breites Spektrum an Geräuschen und Frequenzzusammensetzungen abzudecken, wurden Aufnahmen für Musik, für Sprache, für unterschiedliche Geräusche sowie für eine Kombination dieser benötigt. Konkret wurde dafür die Tonspuren verschiedener Youtube-Videos von Hintergrundgeräuschen und Musik, als Hintergrundgeräusch über die Testsets gelegt.

4 Experimente

4.1 Allgemeine Experimente

Bevor Experimente zu verschiedenen Methoden durchgeführt werden konnten, mussten natürlich erst die optimalen Parameter für die Aufnahme von Audiodaten gefunden werden. Die meisten Parameter für einen Audiostream mittels PyAudio sind durch die Soundkarte vorgegeben. Beispiele dafür sind die Samplingrate (44100 Hz bei der hier benutzten Soundkarte) sowie die Anzahl der Channel. Der einzige Parameter, der beeinflussbar ist, ist die Anzahl der Frames pro Buffer. Dieser legt fest, wie oft die Callback Funktion pro Sekunde aufgerufen wird. Diese Anzahl lässt sich durch folgende Formel berechnen.

$$\text{Callbackaufrufe pro Sekunde} = \frac{\text{Samplingrate}}{\text{Frames pro Buffer}} Hz$$

Ausserdem wird dadurch schon eine erste rechteckige Fensterung vorgegeben. Die Länge der Frame in *ms* lässt sich wie folgt berechnen:

$$\text{Fensterlänge} = \frac{1000}{\frac{\text{Samplingrate}}{\text{Frames pro Buffer}}} ms$$

Logischerweise erhöht sich der Berechnungsoverhead je öfter die Callbackfunktion pro Sekunde aufgerufen wird. Daraus folgt, dass eine höhere Anzahl an Frames pro Buffer weniger Rechenleistung benötigt. Wie sich zeigte, funktioniert das gut bis zu 512 Frames pro Buffer. Bei einer Anzahl von 1024 kam es dann zu einem Input Overflow, es konnten also nicht mehr alle Frames an die Callbackfunktion übergeben werden, womit ein Informationsverlust einhergeht. Somit wurde für eine möglichst optimale Berechnung ein Wert von 512 für Frames pro Buffer gewählt. Daraus folgen folgende Werte für die Anzahl der Callbackaufrufe und die Fensterlänge:

$$\text{Callbackaufrufe pro Sekunde} = \frac{44100\ Hz}{512} = 86,133\ Hz$$

$$\text{Fensterlänge} = \frac{1000}{86,133\ Hz} = 11,6\ ms$$

Da dies eine eher kurze Fensterlänge ist, konnte der Rechenoverhead nochmals reduziert werden, indem jeweils zwei Buffer zusammengefasst wurden. Es wurde also immer abwechselnd einmal nur die eingehenden Daten gespeichert und einmal die im vorherigen Aufruf gespeicherten Daten mit den neuen eingehenden Daten zu einem Frame zusammengefasst und verarbeitet. Dadurch ergibt sich eine Fensterlänge von ca. 23 *ms*

4.2 FFT mit lienarer Fensterung

Bevor verschiedene Methoden und Verfahren gegeneinander verglichen werden können, müssen zuerst für jedes Verfahren die optimalen Parameter bestimmt werden. Folgende Parameter wurden ausgiebig getestet, bevor ein Modell pro Sound mit den optimalen Parametern berechnet wurde:
* Anzahl der Frames im Modell
* Anzahl der weggelassenen Werte pro Frame
* Auswahlkriterium für das Modell (meiste Matches, bester Score)

4.2.1 Modell A: einfacher Pfeiffton

Intuitiv würde für ein Modell mit einem einfachen Pfeiffton nur ein Frame Sinn machen. Da die Aufnahmen, mit denen trainiert wurden, aber nicht nur den Pfeiffton, sondern auch noch Stille beziehungsweise Hintergrundgeräusche vor und nach dem Pfeifen enthält, müssen diese auch beachtet werden. Daher ergibt sich nach dieser Überlegung ein Optimum an drei Frames für ein Geräusch mit nur einer Frequenzverteilung. Ob dies auch in der Praxis die beste Anzahl an Frames ist, wird im Folgenden mit Experimenten überprüft.

4.2.1.1 Auswahlkriterium: Best Score

In den ersten beiden Diagrammen ist die Auswertung der Modells mit 3, 4 und 5 Frames mit jeweils unterschiedlichen Anzahlen an weggelassener Werte pro Frame zu sehen. Als Auswahlkriterium wurde jeweils der beste Score gewählt. Wie erwartet, nimmt die Erkennungsrate mit zunehmender Anzahl an weggelassenen Werten eher zu. In Abbildung 4.1 ist gut zu sehen, dass die Erkennungsrate für ein Modell mit 3 Frames bis zu einer Anzahl von 40 weggelassenen Werten kontinuierlich ansteigt. Werden mehr Werte pro Frame weggelassen, wird die Erkennungsrate wieder schlechter, was allerdings nicht verwunderlich ist. Da jeder Frame nur aus 64 Werten besteht, fallen dort einfach zu viele Informationen weg, um noch sicher erkennen zu können.
In Abbildung 4.2 ist gut erkennbar, dass mit steigender Anzahl an weggelassenen Werten die Anzahl der False Positivs ansteigt. Allerdings fällt auch auf, dass bei einer größeren Anzahl Frames pro Modell die False Positivs vernachlässigt werden können. Lediglich bei 3 Frames spielen sie eine Rolle, wobei selbst bei 56 weggelassenen Frames die 6 False Positivs im Vergleich zu den 50 Testgeräuschen nur 12 % betragen.
Aus diesen Modells wurde dann einerseits das Modell mit 3 Frames und 40 weggelassenen Werten mit einer Erkennungsrate von 98 % als das beste Modell ausgewählt. Dieses Modell hatte in den Tests 5 False Positivs, also eine False Positiv Rate von 10 %. Andererseits wurde auch das Modell mit 4 Frames und 8 weggelassenen Werten ausgewählt, da es eine Erkennungsrate von 90 % und keine False Positivs hat.

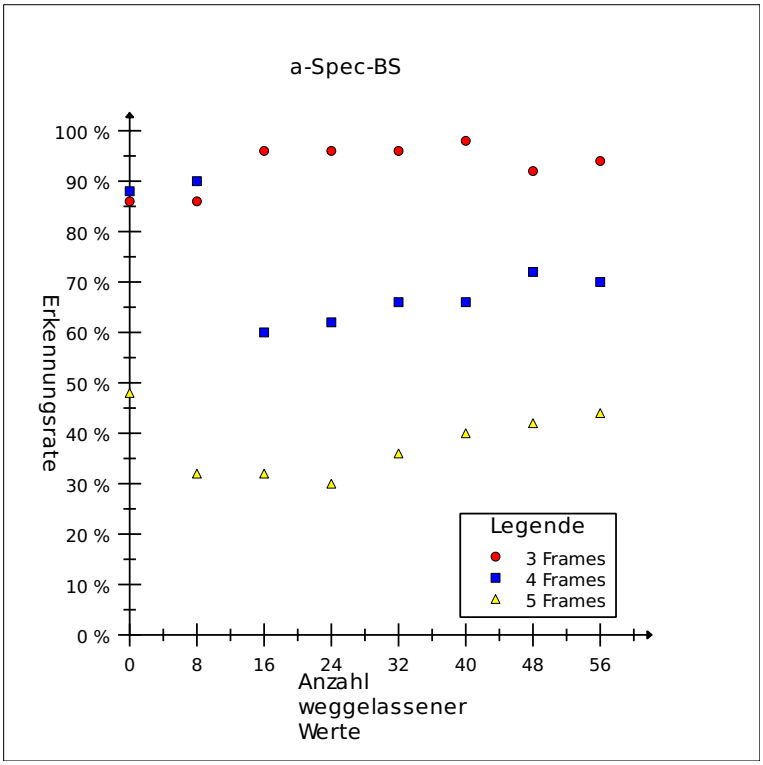

Abbildung 4.1: Spektrum Best Score: Weggelassene Werte zu Erkennungsrate

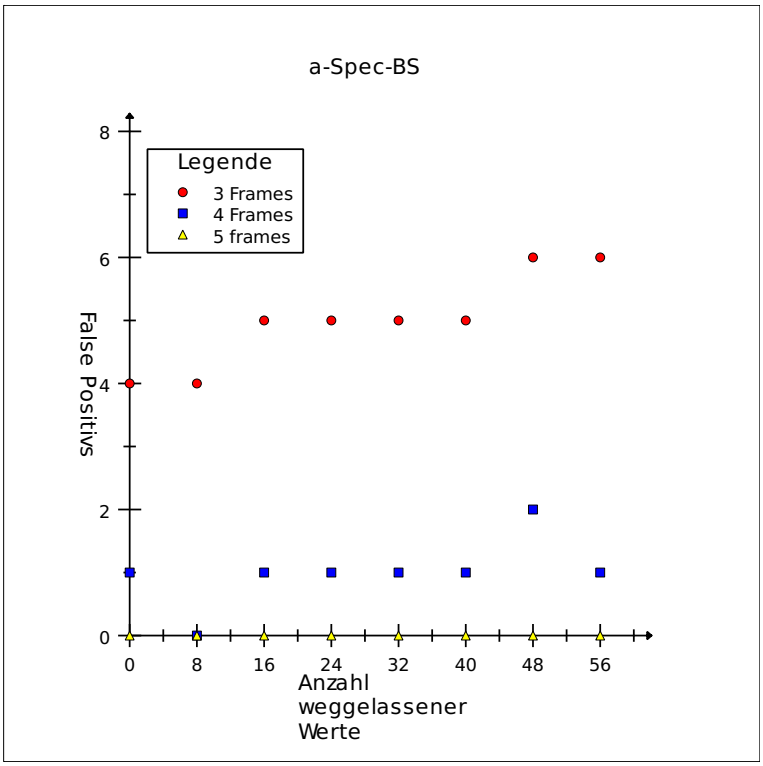

Abbildung 4.2: Spektrum Best Score: Weggelassene Werte zu False Positivs

4.2.1.2 Auswahlkriterium: Most Matches

In den folgenden beiden Diagrammen sind die gleichen Experimente wie für das Auswahlkriterium Best Score zu sehen, allerdings war das Auswahlkriterium bei diesen Modells nicht der beste Score, sondern die meisten Matches.
Wie in Abbildung 4.3 gut zu sehen ist, ist die Erkennungsrate für 3 und 5 Frames leicht höher als beim Auswahlkriterium Best Score und für 4 Frames deutlich höher. In Abbildung 4.4 ist zu erkennen, dass die Anzahl der False Positivs leicht niedriger als beim Auswahlkriterium Best Score ist. Sollte sich dies auch bei den anderen Modells zeigen, so muss die Berechnung des Scores eventuell nochmal überdacht werden.
Hier wurde ein Modell ausgewählt, um später nochmal ausgiebig getestet zu werden. Das Modell mit 4 Frames und 40 weggelassenen Werten hat nicht nur die höchste Erkennungsrate, sondern auch keine False Positivs.

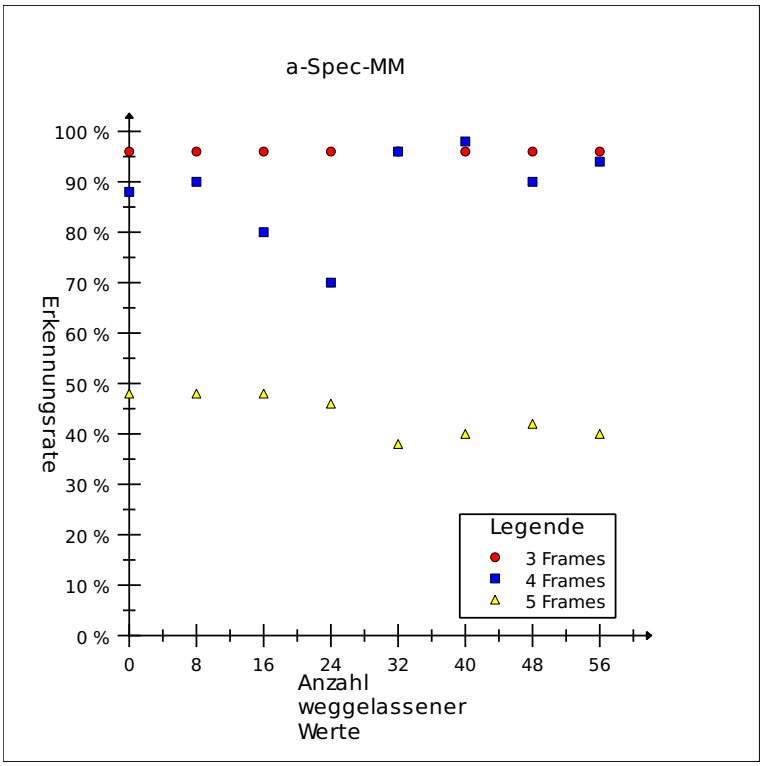

Abbildung 4.3: Spektrum Most Matches: Weggelassene Werte zu Erkennungsrate

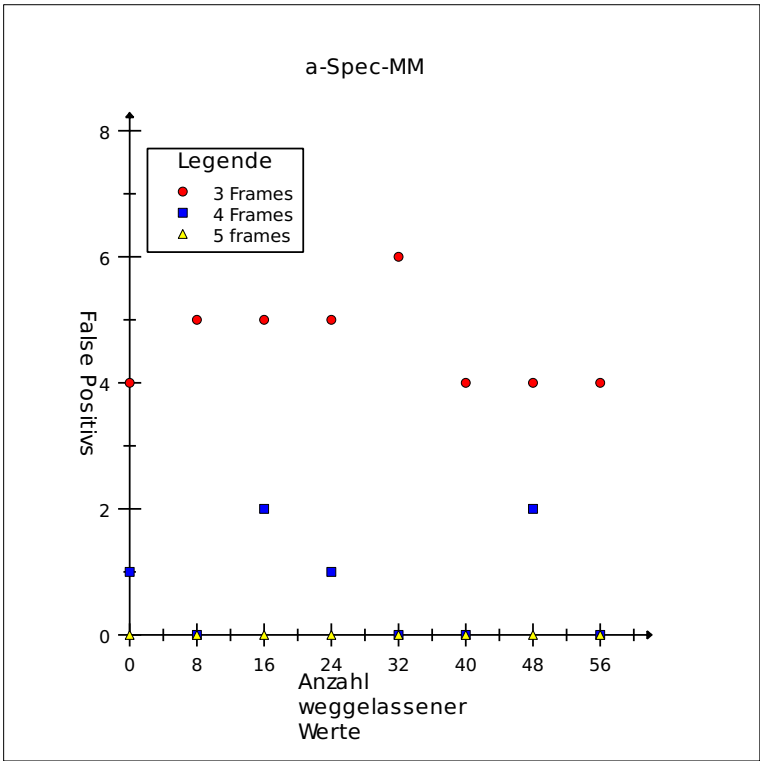

Abbildung 4.4: Spektrum Most Matches: Weggelassene Werte zu False Positivs

4.2.2 Modell B: Schließen einer Tür

Wie sich während der Aufnahmen zeigte, besteht das Geräusch des Türschließens aus zwei Geräuschen. Zuerst knallt die sogenannte Falle gegen den Rahmen und wird dann ins Schloss gedrückt. Danach knallt die Tür an den Rahmen, während die Falle wieder hervorschnappt und einrastet. Somit sind zwei Geräusche sowie je nach Geschwindigkeit eine kleine Pause dazwischen zu hören. Intuitiv wäre also eine Anzahl von vier bis fünf Frames auszuwählen. Um sicher zu gehen, dass mehr Frames hier nicht vielleicht doch besser sind, wurde das Modell mit 4, 5, 6, 7 und 8 Frames trainiert.

4.2.2.1 Auswahlkriterium: Best Score

Auch hier ist wieder, wie beim vorherigen Modell, die Auswertung der Erkennungsrate, im Vergleich zu der Anzahl der weggelassenen Werte unter dem Auswahlkriterum Best Score zu sehen. Wie erwartet, nimmt auch hier die Erkennungsrate mit zunehmender Anzahl an weggelassenen Werten eher zu. In Abbildung 4.5 ist gut zu sehen, dass die Erkennungsrate für ein Modell mit 4 Frames bis zu einer Anzahl von 48 weggelassenen Werten beinahe konstant ist. Bei 4 und 5 Frames sieht die Erkennungsrate für wenig weggelassene Werte beinahe gleich aus, erst bei mehreren weggelassenen Werte unterscheiden sie sich. Erstaunlicherweise ist das Modell mit 5 Frames für 24, 32, 40 und 56 weggelassenen Werten deutlich schlechter als das Modell mit 6 Frames, für 48 weggelassene Werte allerdings deutlich besser ist. Die Modells mit 7 und 8 Frames sind dahingegen durchweg schlecht

und nicht zu gebrauchen.
Überraschenderweise wurde bei keinem einzigen Modell False Positivs erkannt. Vermutlich liegt das daran, dass der Geräuscherkenner bei diesem Verfahren nur für Modells mit wenigen Frames, wie zum Beispiel drei, False Positivs erkennt. Sobald ein Modell 4 oder mehr Frames hat, traten bisher entweder keine False Positivs auf oder waren vernachlässigbar.
Aus diesen Modells wurden mehrere verschiedene ausgewählt, um sie nochmals genauer anzuschauen. Es wurden von den Modells mit 4 Frames sowohl das ohne weggelassene Werte, sowie das mit 48 weggelassenen Werten ausgewählt, um zu prüfen, ob die Anzahl der weggelassenen Werte bei Testsets mit Hintergrundgeräuschen einen Unterschied macht. Beide haben eine Erkennungsrate von 100 %. Außerdem wurde das Modell mit 5 Frames und 48 weggelassenen Werten mit einer Erkennungsrate von 80 %, sowie das Modell mit 6 Frames und 56 weggelassenen Werten mit einer Erkennungsrate von 86 % ausgewählt.

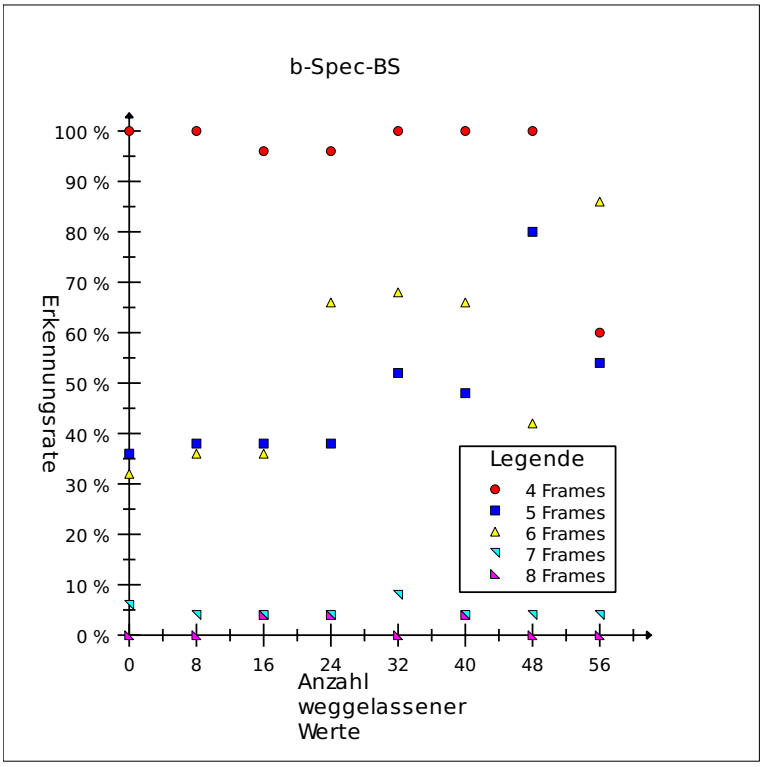

Abbildung 4.5: Spektrum Best Score: Weggelassene Werte zu Erkennungsrate

4.2.2.2 Auswahlkriterium: Most Matches

Wie in Abbildung 4.6 auffällt, sind die Erkennungsraten für die Modells mit 4 Frames beim Auswahlkriterium Most Matches identisch mit denen beim Auswahlkriterum Best Score. Für die Modells mit 5 Frames liegen die Erkennungsraten vor allem bei mehr weggelassenen Werten eindeutig höher. Bei den Modells mit 6 Frames sind einge sehr ähnlich wie beim Auswahlkriterium Best Score, einige aber auch deutlich schlechter. Die

Modells mit 7 und 8 Frames können auch hier ignoriert werden, da sie sehr schlecht sind. Wie auch schon beim Auswahlkriterium Best Score, wurden hier keine False Positivs erkannt.
Hier wurden keine Modells nochmals genauer angeschaut, da die Ergebnisse ähnlich wie beim Auswahlkriterium Best Score waren.

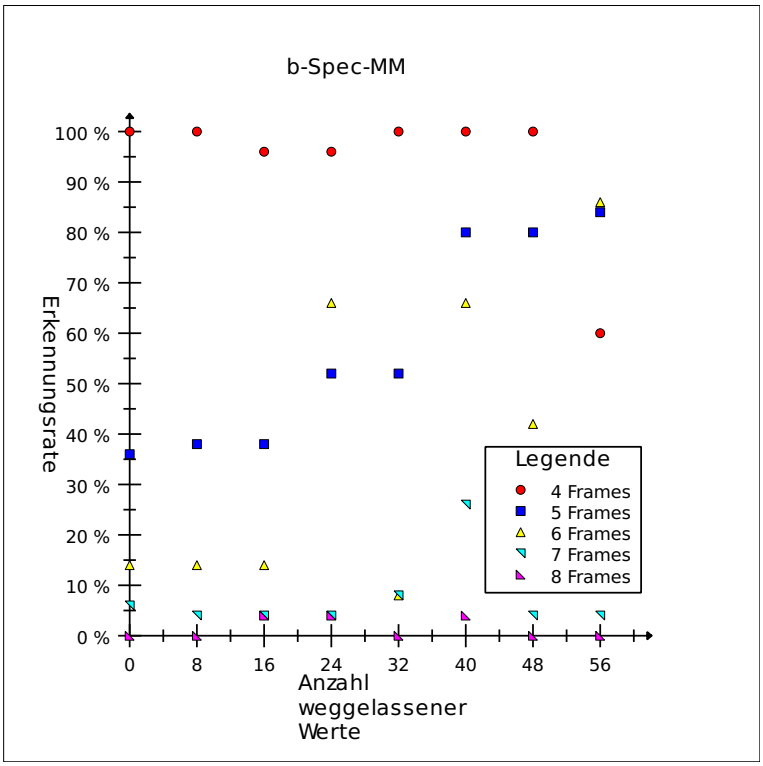

Abbildung 4.6: Spektrum Most Matches: Weggelassene Werte zu Erkennungsrate

4.2.3 Modell C: Ofentimer

Das hier aufgenommene Geräusch ist ein Ofentimer, der den Ofen nach einer vorher eingestellten Zeit abschaltet und dies durch ein Glockenklingen anzeigt. Das Geräusch ähnelt also einem einfachen Pfeifen, nur dass anstatt dem Pfeifen ein “Pling” zu hören ist. Dementsprechend wurden auch hier Modells mit 3, 4 und 5 Frames trainiert.

4.2.3.1 Auswahlkriterium: Best Score

In Abbildung 4.7 fällt zuerst auf, dass die Erkennungsraten recht unterschiedlich aussehen. Für die Modells mit 3 Frames steigt die Erkennungsrate ab 32 weggelassenen Werten bis zu 48 weggelassenen Werten stark an. Für 56 weggelassene Werte ist sie allerdings, wie auch für 4 und 5 Frames schlecht. Sowohl für die Modells mit 4 Frames, als auch für die mit 5 Frames gilt, dass sie sowohl für wenig, als auch für viele weggelassene Werte eher schlecht sind. Nur im mittleren Bereich, bei 4 Frames von 16 bis 40, bei 5 Frames von 24 bis

40 weggelassenen Werten, ist die Erkennungsrate etwas höher. Ihr jeweiliges Maximum erreicht sie bei 4 Frames und 24 weggelassenen Werten mit 74 % und bei 5 Frames und 24 weggelassenen Werten bei 36 %.
Wie schon bei Modell B, wurden auch hier keine False Positivs erkannt.
Aus diesen Modells wurde nur das Modell mit 3 Frames, 48 weggelassenen Werten und einer Erkennungsrate von 98 % ausgewählt.

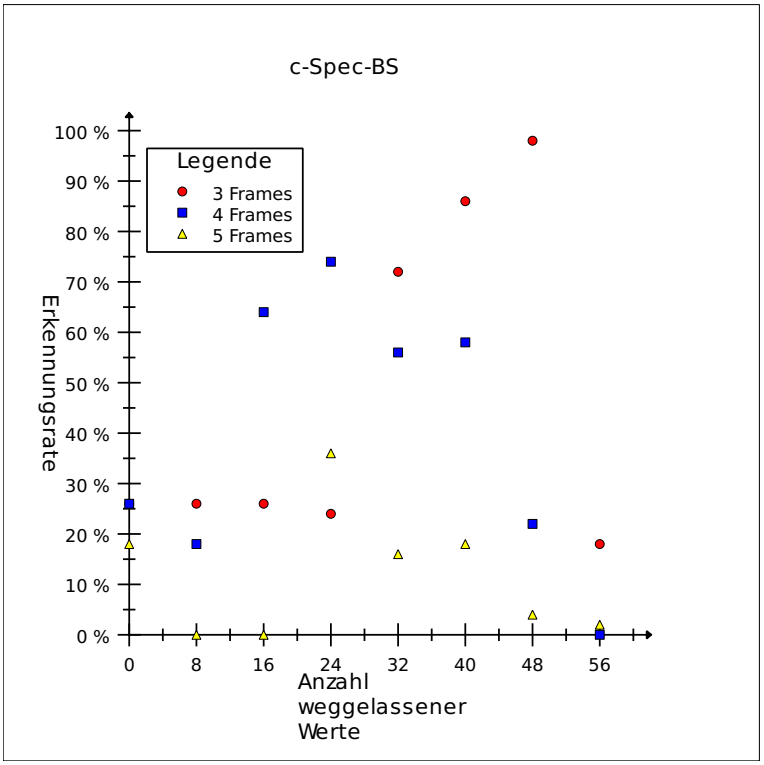

Abbildung 4.7: Spektrum Best Score: Weggelassene Werte zu Erkennungsrate

4.2.3.2 Auswahlkriterium: Most Matches

Wie in Abbildung 4.8 gut zu sehen ist, ähneln sich die Erkennungsraten der beiden unterschiedlichen Auswahlkriterien sehr. Für das Auswahlkriterium Most Matches ist die Erkennungsrate teilweise leicht besser, wie zum Beispiel für das Modell mit 4 Frames und 40 weggelassenen Werten. Für die meisten Modells sind die Erkennungsraten allerdings gleich.
Auch hier wurden keine False Positivs erkannt.
Da die Erkennungsraten nahezu identisch waren, wurde hier kein Modell ausgewählt.

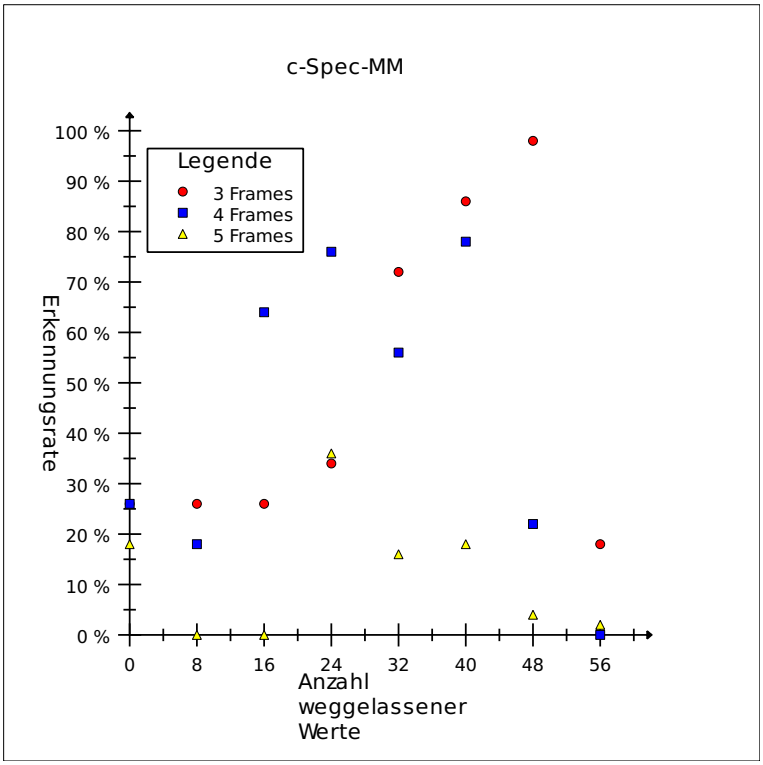

Abbildung 4.8: Spektrum Most Matches: Weggelassene Werte zu Erkennungsrate

4.2.4 Modell D: pfeifen von zwei Tönen

Bei diesem Modell wurde das Pfeifen zweier aufeinanderfolgender, unterschiedlicher Töne aufgenommen. Es wurde versucht, keine Pause zwischen den Tönen zu lassen. Der Argumentation der bisherigen Experimente folgend, müssten für dieses Modell sowohl 4, 5 als auch 6 Frames trainiert werden. Da die bisherigen Experimente allerdings zeigten, dass weniger Frames meistens besser sind, wurde die Modells mit 3, 4 und 5 Frames trainiert.

4.2.4.1 Auswahlkriterium: Best Score

In Abbildung 4.9 ist zu sehen, dass die Erkennungsraten sowohl für wenig sowie für viele weggelassenen Werte etwas besser ist, als im Bereich dazwischen. Insgesamt sind die Schwankungen allerdings kleiner als zum Beispiel beim Ofentimer. Es fällt auf, dass die Schwankungen um so größer werden, je mehr Frames die Modells haben. Für 3 Frames liegt die Differenz zwischen höchster und niedrigster Erkennungsrate nur bei 4 Prozentpunkten, bei 4 Frames schon bei 18 Prozentpunkten und bei 5 Frames bei 26 Prozentpunkten.
Wird nur Abbildung 4.9 betrachtet, könnte der Trugschluss aufkommen, die Modells mit 3 Frames wären die besten. Allerdings zeigt Abbildung 4.10, dass die False Positiv Raten für 3 Frames recht hoch liegen. Es ist zu sehen, dass die False Positivs umso weniger werden, je mehr Werte weggelassen werden, obwohl sich die Erkennungsrate kaum ändert. Dementsprechend müssten die Modells mit mehr weggelassenen Werten also besser sein

als diejenigen, bei denen wenige Werte weggelassenen wurden. Für die Modells mit 4 und 5 Frames gibt es, außer einem minimalen Ausrutscher bei 4 Frames und 0 beziehungsweise 8 weggelassenen Werten, keine False Positivs.
Aus diesen Modells wurden einmal das Modell mit 3 Frames und keinen weggelassenen Werten mit einer Erkennungsrate von 100 % und 23 False Positivs, sowie das Modell mit 3 Frames und 56 weggelassenen Werten mit einer Erkennungsrate von 98 % und nur 12 False Positivs ausgewählt, um diese mit Hintergrundgeräuschen vergleichen zu können. Außerdem wurde das Modell mit 4 Frames und 56 weggelassenen Werten mit einer Erkennungsrate von 90 % und keinen False Positivs ausgewählt.

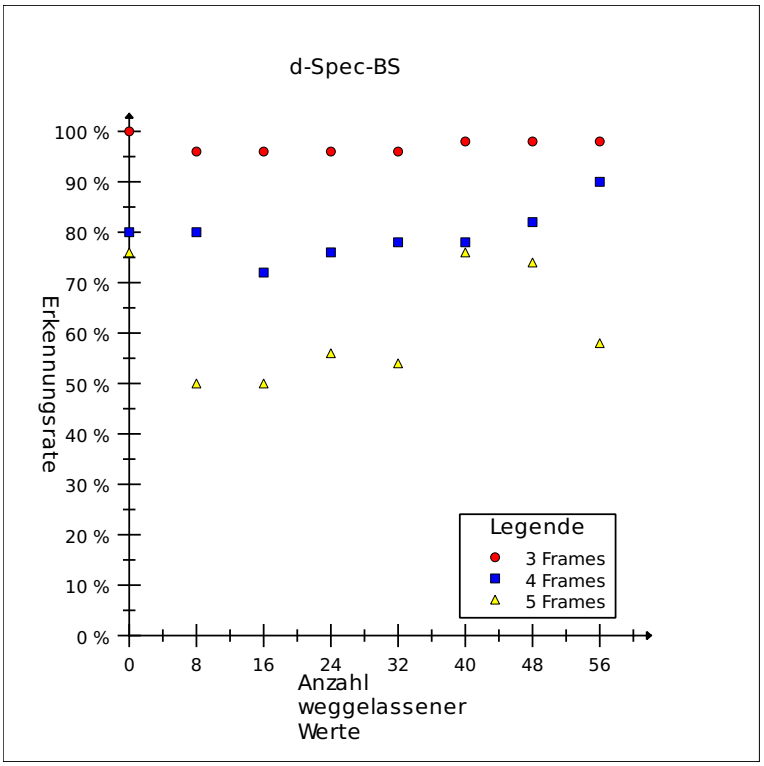

Abbildung 4.9: Spektrum Best Score: Weggelassene Werte zu Erkennungsrate

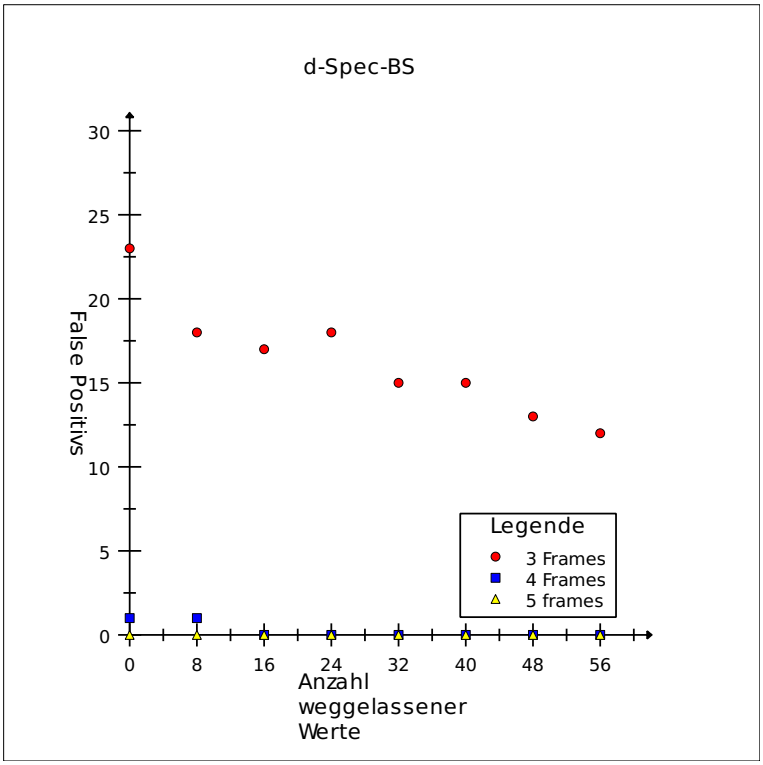

Abbildung 4.10: Spektrum Best Score: Weggelassene Werte zu False Positivs

4.2.4.2 Auswahlkriterium: Most Matches

Wie in Abbildung 4.11 deutlich zu sehen ist, ist die Erkennungsrate für 4 und 5 Frames höher, als beim Auswahlkriterium Best Score. Die Modells mit 4 Frames sind fast genau so gut, wie die Modells mit 3 Frames.
Abbildung 4.12 zeigt, dass die Modells mit 3 Frames recht viele False Positiv erkennen, die Anzahl liegt immer zwischen 18 und 23. Die Modells mit 4 oder 5 Frames haben keine oder maximal ein False Positiv erkannt.
Hier wurden einmal das Modell mit 3 Frames und 40 weggelassenen Werten, mit einer Erkennungsrate von 100 % und 18 False Positivs, sowie das Modell mit 4 Frames und 40 weggelassenen Werten, mit einer Erkennungsrate von 98 % und keinen False Positivs, ausgewählt.

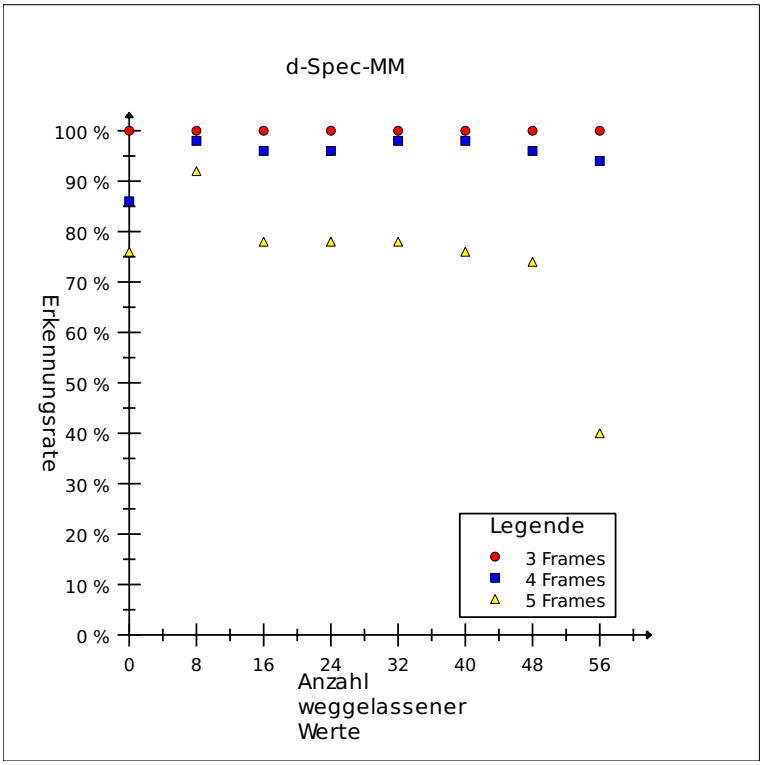

Abbildung 4.11: Spektrum Most Matches: Weggelassene Werte zu Erkennungsrate

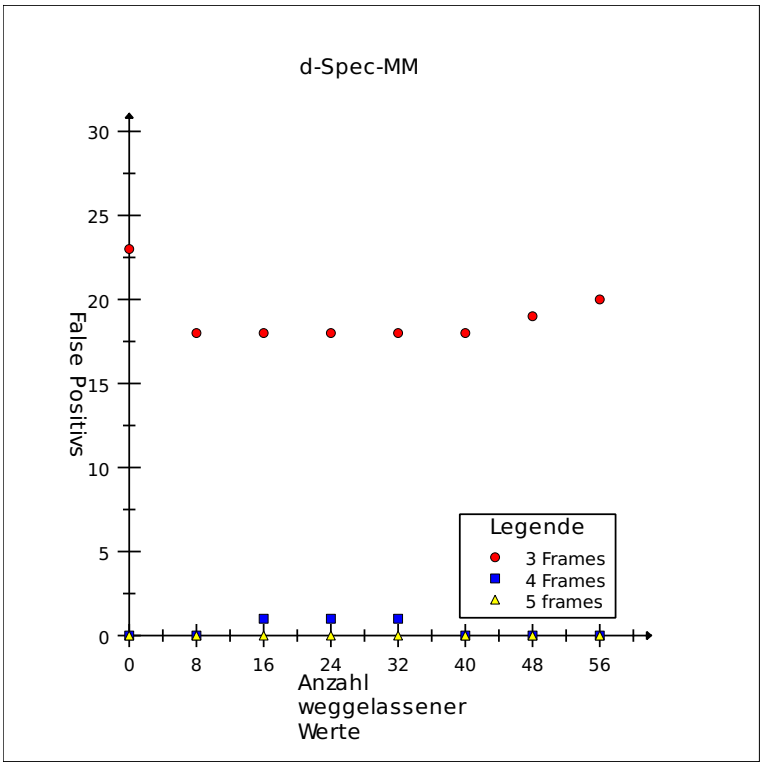

Abbildung 4.12: Spektrum Most Matches: Weggelassene Werte zu False Positivs

4.2.5 Modell E: Türklingel

Bei diesem Modell wurde das kurze Klingeln einer Türklingel mit 3, 4 und 5 Frames trainiert.

4.2.5.1 Auswahlkriterium: Best Score

In Abbildung 4.13 fällt zuerst auf, dass die Erkennungsrate für die Modells mit 4 und 5 Frames und 40 oder mehr weggelassenen Werten, immer 0 % ist. Für weniger weggelassene Werte sind die Modells mit 4 Frames fast genau so gut wie die mit 3 Frames. Die Modells mit 5 Frames steigern sich bis zu 32 weggelassenen Werten, bis sie, wie oben gesagt, gar nichts mehr erkennen.
False Positivs wurden bei diesen Modells keine erkannt.
Da mehr weggelassene Werte vermutlich bessere Erkennungsraten liefern, werden von diesen Modells sowohl das mit 3, als auch mit 4 Frames und jeweils 24 weggelassenen Werten ausgewählt, um sie später nochmals ausführlicher zu testen.

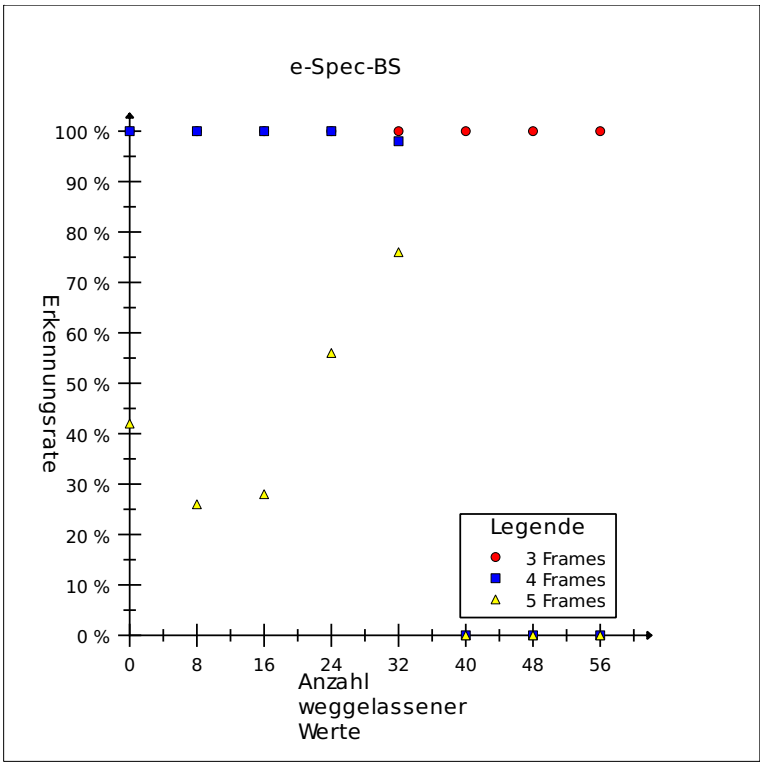

Abbildung 4.13: Spektrum Best Score: Weggelassene Werte zu Erkennungsrate

4.2.5.2 Auswahlkriterium: Most Matches

In Abbildung 4.14 ist zu sehen, dass die Ergebnisse für 3 und 5 Frames genau gleich wie beim Auswahlkriterium Best Score sind. Bei den Modells mit 4 Frames fällt auf, dass die Modells mit weniger weggelassenen Werten etwas besser sind, als beim Auswahlkriterium Best Score. Allerdings sind diese auch noch nicht wirklich gut.

False Positivs wurden auch hier keine erkannt.
Da die Ergebnisse nahezu identisch wie bei dem Auswahlkriterium Best Score sind, wurde hier kein Modell ausgewählt, um es nochmal genau zu testen.

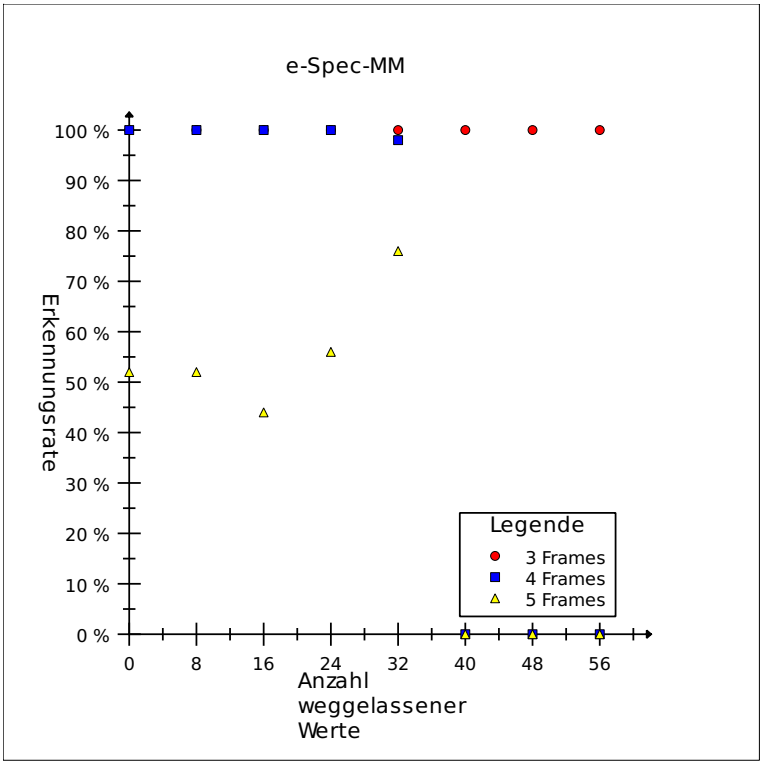

Abbildung 4.14: Spektrum Most Matches: Weggelassene Werte zu Erkennungsrate

4.3 Cepstrum mit lienarer Fensterung ohne Liftering

Auch bei diesem Verfahren wurden zunächst für jedes Geräusch das optimale Modell gesucht, bevor diese gegeneinander abgewogen werden können. Die Parameter sind wie oben:
* Anzahl der Frames im Modell
* Anzahl der weggelassenen Werte pro Frame
* Auswahlkriterium für das Modell (meiste Matches, bester Score)
Es wurden immer genau die gleichen Parameter wie beim Verfahren FFT mit linearer Fensterung ausgewählt, nur die Merkmalsextraktion wurde anders berechnet. Insgesamt zeigt sich, dass dieses Verfahren nur bedingt geeignet ist, Geräusche stabil zu erkennen.

4.3.1 Modell A: einfacher Pfeiffton

Wie schon zuvor, wurden Modells mit 3, 4 und 5 Frames und unterschiedlicher Anzahl an weggelassenen Werten berechnet.

4.3.1.1 Auswahlkriterium: Best Score

In Abbildung 4.15 fällt auf, dass die Erkennungsraten unabhängig von der Anzahl der weggelassenen Werte ist. Da die Erkennungsrate für 3 Frames bei 100 % liegt, wäre zu vermuten, dass diese die besten Modells sind. Allerdings fällt bei einem Blick auf Abbildung 4.16 auf, dass diese Modells mit 28 False Positivs bei 50 getesteten Geräuschen eine hohe Fehlerrate haben.
Aus diesen Modells wurden keine ausgewählt, da das Auswahlkriterium Most Matches bessere Ergebnisse erzielte.

4.3.1.2 Auswahlkriterium: Most Matches

Wird nur Abbildung 4.17 betrachtet, so scheinen die Ergebnisse für beide Auswahlkriterien gleich zu sein, in Abbildung 4.18 ist allerdings zu erkennen, dass die Modells mit 3 Frames beim Auswahlkriturium Most matches nur 13 False Positivs anstatt, wie beim Auswahlkriterium Best Score, 28 haben.
Es wurden dann das Modell mit 3 Frames und keinen weggelassenen Werten, sowie das mit 3 Frames und 56 weggelassenen Werten ausgewählt, um sie nochmal genauer zu testen.

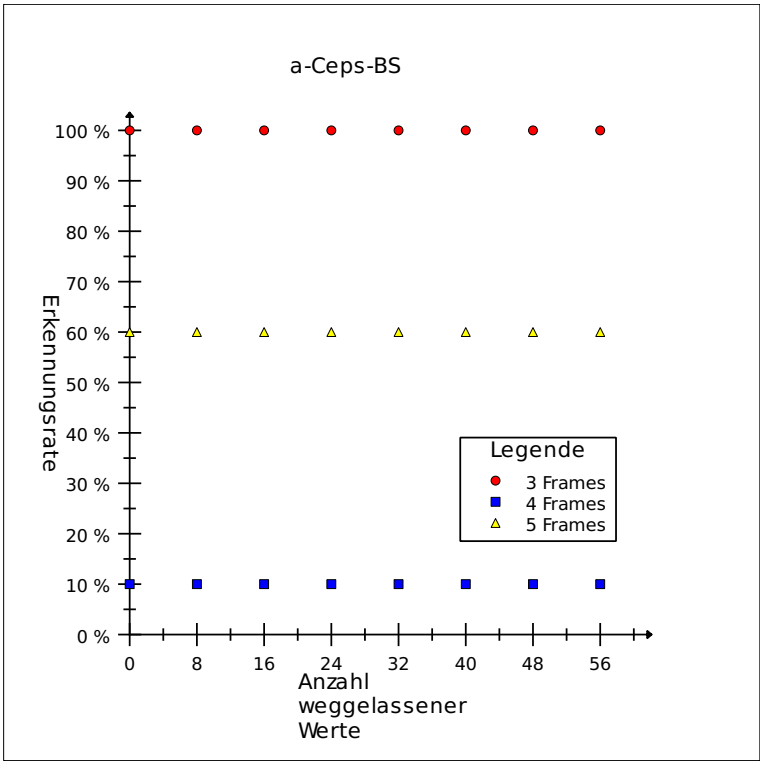

Abbildung 4.15: Cepstrum Best Score: Weggelassene Werte zu Erkennungsrate

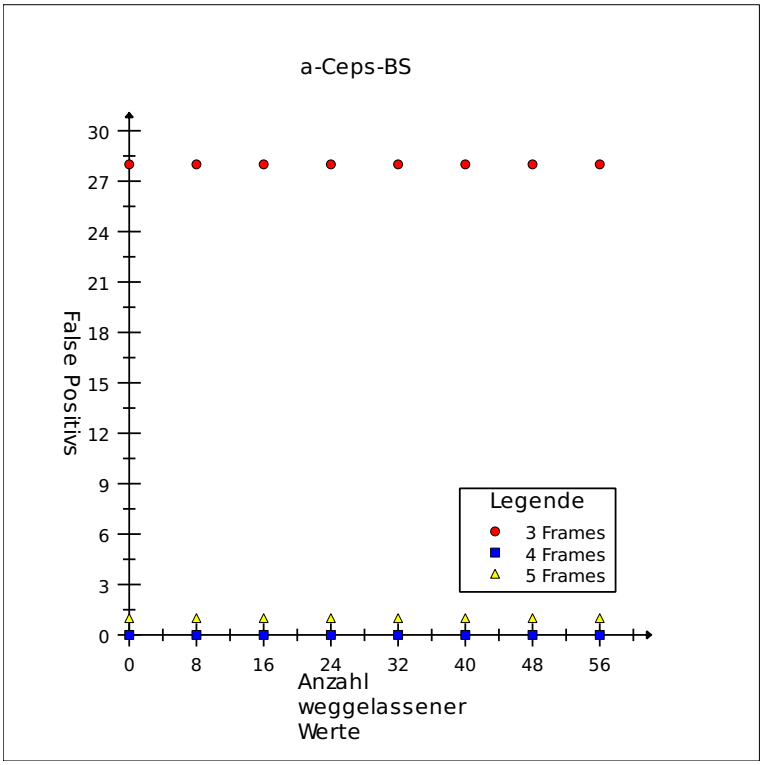

Abbildung 4.16: Cepstrum Best Score: Weggelassene Werte zu False Positivs

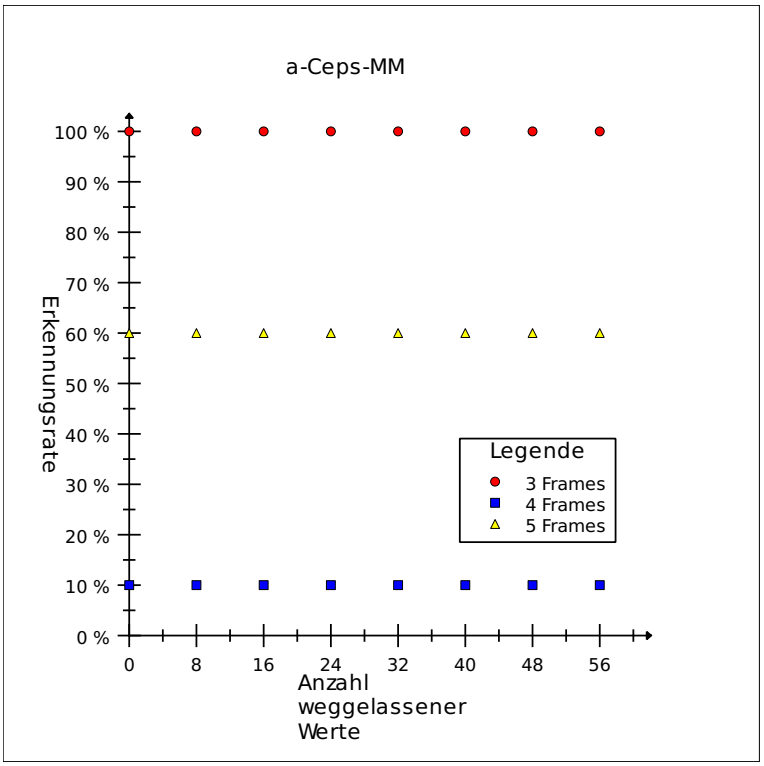

Abbildung 4.17: Cepstrum Most Matches: Weggelassene Werte zu Erkennungsrate

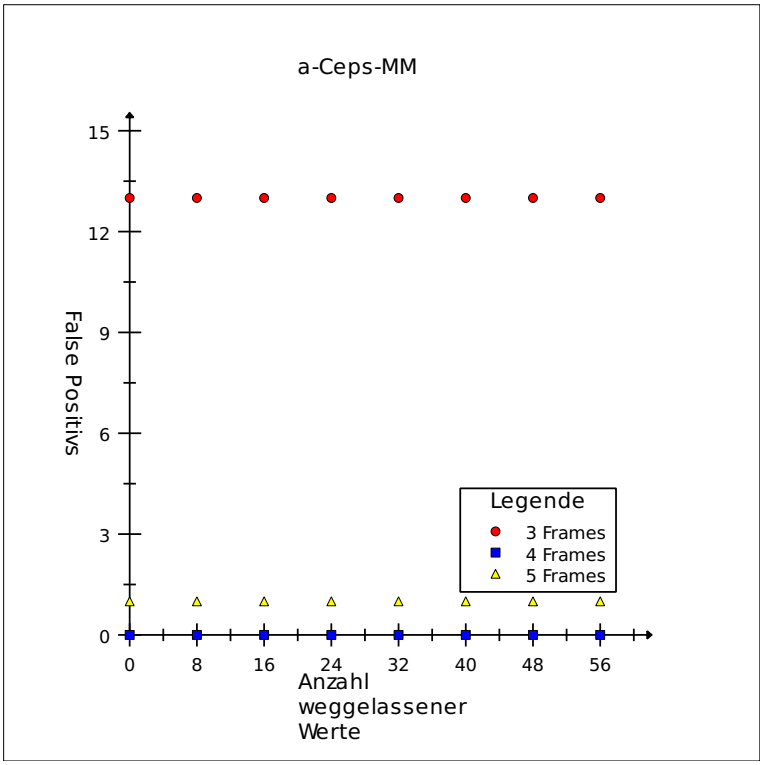

Abbildung 4.18: Cepstrum Most Matches: Weggelassene Werte zu False Positivs

4.3.2 Modell B: Schließen einer Tür

Wie schon beim Verfahren FFT mit linearer Fensterung, wurden Modells mit 4, 5, 6, 7 und 8 Frames und unterschiedlicher Anzahl an weggelassenen Werten berechnet.

4.3.2.1 Auswahlkriterium: Best Score

In Abbildung 4.19 ist zu sehen, dass die Erkennungsrate für die Modells mit 4 Frames bei 100 % liegt. Für die Modells mit 5 und 7 Frames wurde nur für keine weggelassenen Werte etwas erkannt, bei den Modells mit 6 Frames nur bei 8 weggelassenen Frames. Für 8 Frames allerdings wurde für alle Anzahlen an weggelassenen Werten eine konstante Erkennungsrate von 40 % geliefert.
Es wurden bei keinen dieser Modells False Positivs erkannt.
Aus diesen Modells wurden von den Modells mit 4 Frames das mit keinen weggelassenen, sowie das mit 56 weggelassenen Werten ausgewählt. Außerdem wurde das Modell mit 5 Frames und keinen weggelassenen Werten, das eine Erkennungsrate von 82 % hat, sowie die Modells mit 8 Frames und keinen beziehungsweise 56 weggelassenen Werten mit jeweils einer Erkennungsrate von 40 %, ausgewählt.

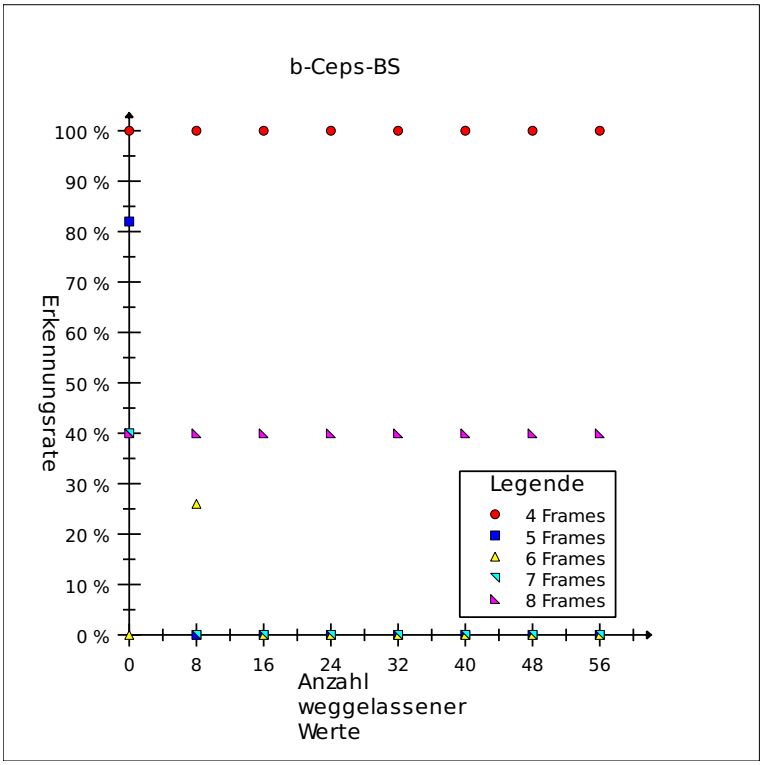

Abbildung 4.19: Cepstrum Best Score: Weggelassene Werte zu Erkennungsrate

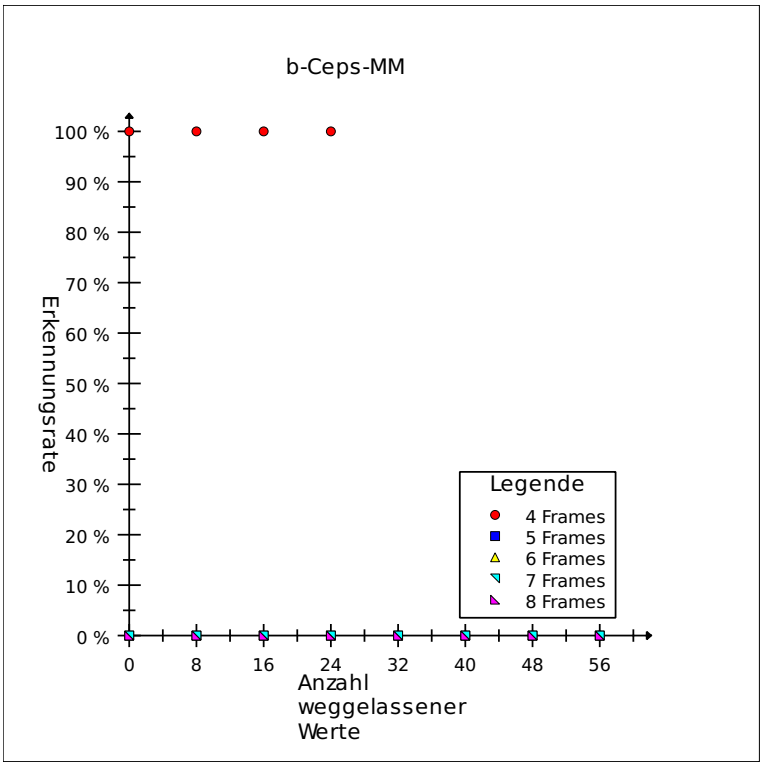

Abbildung 4.20: Cepstrum Most Matches: Weggelassene Werte zu Erkennungsrate

4.3.2.2 Auswahlkriterium: Most Matches

Hier wurden nur für 4 Frames und 0, 8, 16 und 24 weggelassene Werte überhaupt etwas erkannt. In Abbildung 4.20 ist zu sehen, dass diese Modells alle eine Erkennungsrate von 100 % haben. False Positivs wurden bei keinem der Modells erkannt.
Da die Ergebnisse für die Modells, für die es bei diesem Auswahlkriterium überhaupt Ergebnisse gab, genau gleich sind wie beim Auswahlkriterium Best Score, wurden hier keine Modells ausgewählt.

4.3.3 Modell C: Ofentimer

Für dieses Geräusch wurden Modells mit 3, 4 und 5 Frames trainiert, allerdings hat keines dieser Modells irgendetwas erkannt, weswegen hierzu keine Testdaten vorliegen.

4.3.4 Modell D: pfeifen von zwei Tönen

Wie schon zuvor, wurden Modells mit 3, 4 und 5 Frames und unterschiedlicher Anzahl an weggelassenen Werten berechnet.

4.3.4.1 Auswahlkriterium: Best Score

In Abbildung 4.21 ist zu bemerken, dass es nur für das Modell mit 3 Frames und keinen weggelassenen Werten Ergebnisse gab. Dieses Modell hatte eine Erkennungsrate von 100 % und keine False Positivs und wurde deshalb zur erneuten Betrachtung ausgewählt.

4.3.4.2 Auswahlkriterium: Most Matches

Für dieses Auswahlkriterium wurde nichts erkannt.

4.3.5 Modell E: Türklingel

Auch hier wurden Modells mit 3, 4 und 5 Frames berechnet, allerdings wurde bei keinem Modell etwas erkannt.

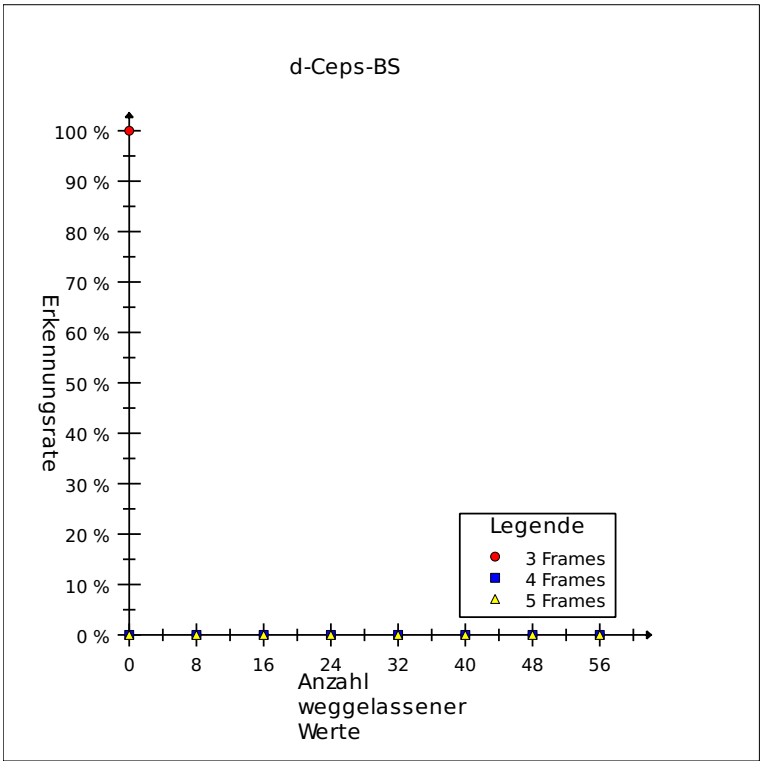

Abbildung 4.21: Cepstrum Best Score: Weggelassene Werte zu Erkennungsrate

4.4 Cepstrum mit lienarer Fensterung mit Liftering

Auch bei diesem Verfahren wurden erstmal für jedes Geräusch das optimale Modell gesucht, bevor diese gegeneinander abgewogen werden können.

4.4.1 Modell A: einfacher Pfeiffton

Wie schon bei den beiden Verfahren zuvor, wurden Modells mit 3, 4 und 5 Frames und unterschiedlicher Anzahl an weggelassenen Werten berechnet.

4.4.1.1 Auswahlkriterium: Best Score

In Abbildung 4.22 ist zu bemerken, dass es nur für die Modells mit 3 Frames und 8, 16, 32 und 40 weggelassenen Werten Ergebnisse gibt. Diese sehen zwar bis auf das Modell mit 32 weggelassenen Werten auf den ersten Blick ganz gut aus, allerdings relativiert das Abbildung 4.23 insofern, dass die Modells eine hohe Anzahl an False Positivs haben. Der Ausreißer bei 40 weggelassenen Frames mit 9142 False Positivs lässt vermuten, dass dieses Verfahren wenig geeignet ist, Geräusche robust zu erkenne.
Aufgrund der hohen False Positiv Rate, selbst das Modell mit der geringsten Rate hat immerhin 106 False Positivs, wurde hier kein Modell ausgewählt.

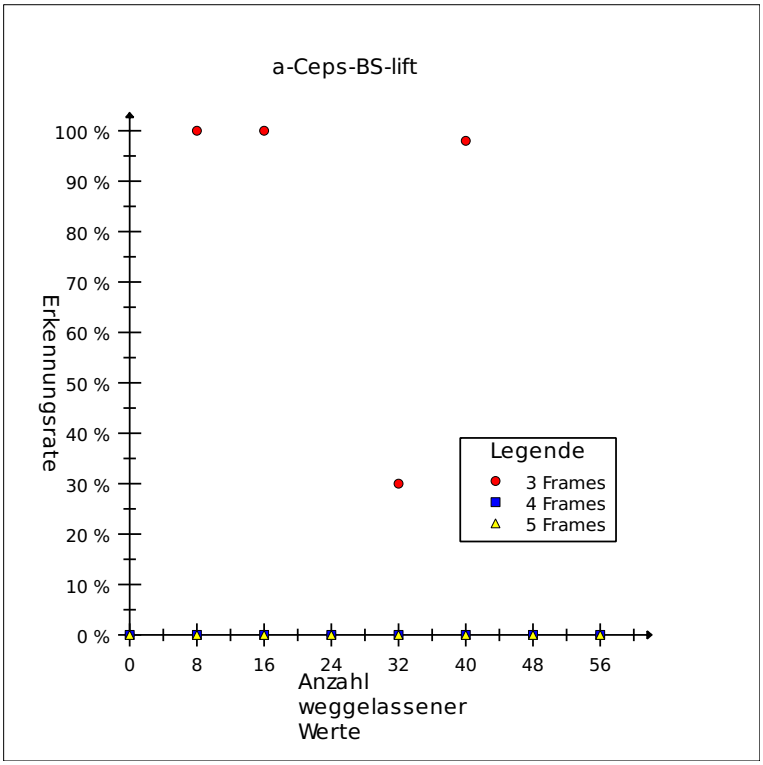

Abbildung 4.22: Cepstrum mit Liftering BS: Weggelassene Werte zu Erkennungsrate

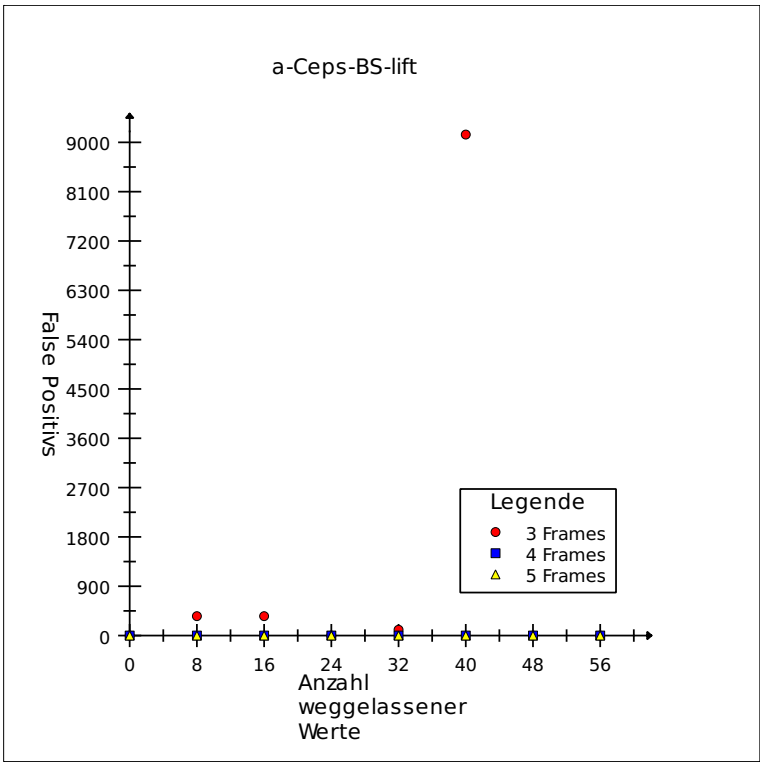

Abbildung 4.23: Cepstrum mit Liftering BS: Weggelassene Werte zu False Positivs

4.4.1.2 Auswahlkriterium: Most Matches

Auch hier wurden nur für vier Modells überhaupt Ergebnisse erzielt, die dann, wie in Abbildung 4.24 zu sehen, auch noch schlechter sind als beim Auswahlkriterium Best Score. Das einzige, was bei diesem Auswahlkriterium besser als beim Auswahlkriterium Best Score ist, ist, dass es in Abbildung 4.25 keinen Ausreißer mit über 9000 False Positivs gibt. Allerdings haben alle Modells mit 3 Frames über 200 False Positivs, was das vierfache der getesteten Geräusche ist.
Auch hier wurden aufgrund der vielen False Positivs keine Modells ausgewählt.

4.4.2 Modell B: Schließen einer Tür

Obwohl hier, wie bei den beiden vorherigen Verfahren, Modells mit 4, 5, 6, 7 und 8 Frames und unterschiedlichen Anzahlen an weggelassenen Werten berechnet wurden, wurde mit keinem der Modells ein Ergebniss erzielt.

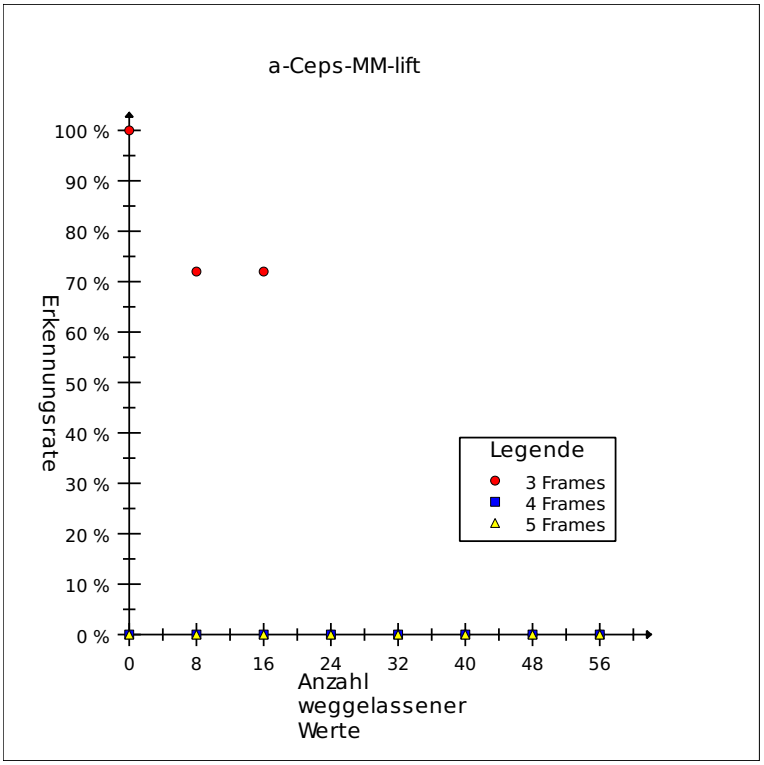

Abbildung 4.24: Cepstrum mit Liftering MM: Weggelassene Werte zu Erkennungsrate

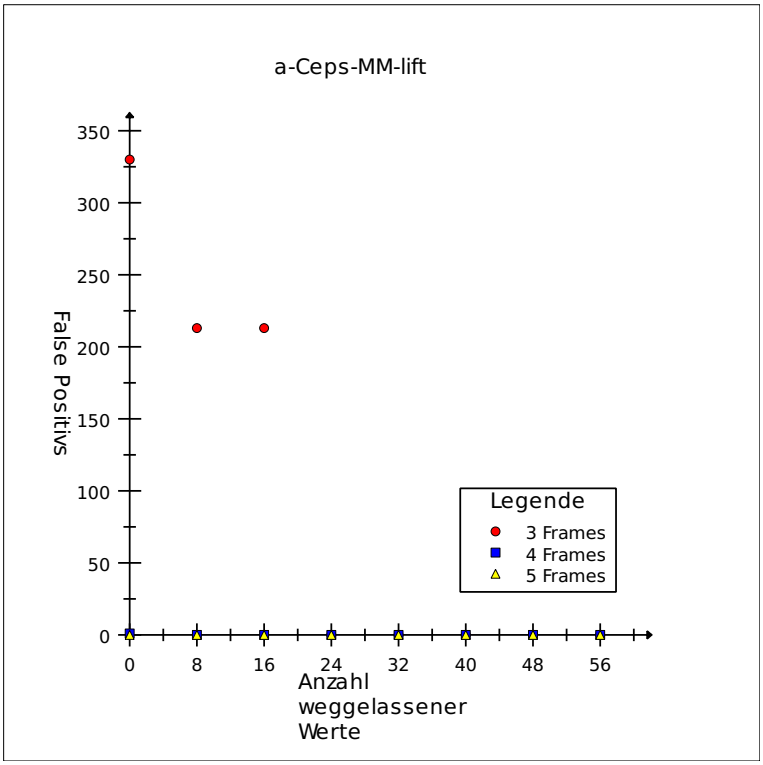

Abbildung 4.25: Cepstrum mit Liftering MM: Weggelassene Werte zu False Positivs

4.4.3 Modell C: Ofentimer

Wie schon bei den beiden Verfahren zuvor, wurden Modells mit 3, 4 und 5 Frames und unterschiedlicher Anzahl an weggelassenen Werten berechnet.

4.4.3.1 Auswahlkriterium: Best Score

In Abbildung 4.26 ist zu bemerken, dass es nur für einen Teil der Modells Ergebnisse gibt. Es fällt auf, dass sowohl für 8 weggelassene Werte, wie auch für 16 weggelassene Werte mit 3, 4 und 5 Frames etwas erkannt wurde. Erstaunlicherweise ist das Modell mit 5 Frames in beiden Fällen mit einer Erkennungsrate von 70 % das beste.
Dank Abbildung 4.27 fällt auf, dass die Modells mit 3 Frames alle über 250 False Positivs haben, was das fünffache der Testgeräusche war. Die Anzahl der False Positivs ist für die Modells mit 5 Frames mit 28 gerade noch vertretbar. Deswegen wurden diese beiden ausgewählt, um sie nochmal genauer zu betrachten.

4.4.3.2 Auswahlkriterium: Most Matches

Wie in Abbildung 4.28 und Abbildung 4.29 zu sehen ist, sind die Ergebnisse genau gleich wie für das Auswahlkriterium Best Score, somit wurden hier nicht nochmal Modells ausgewählt, um sie extra zu testen.

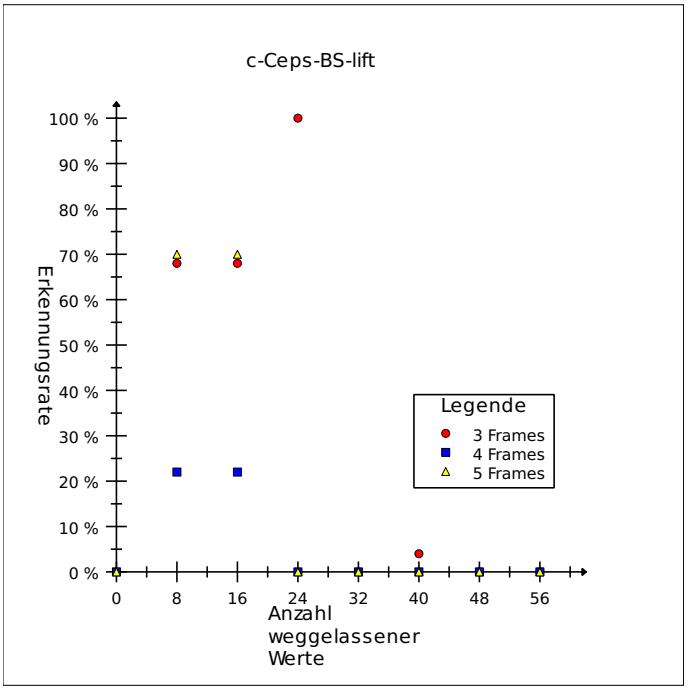

Abbildung 4.26: Cepstrum mit Liftering BS: Weggelassene Werte zu Erkennungsrate

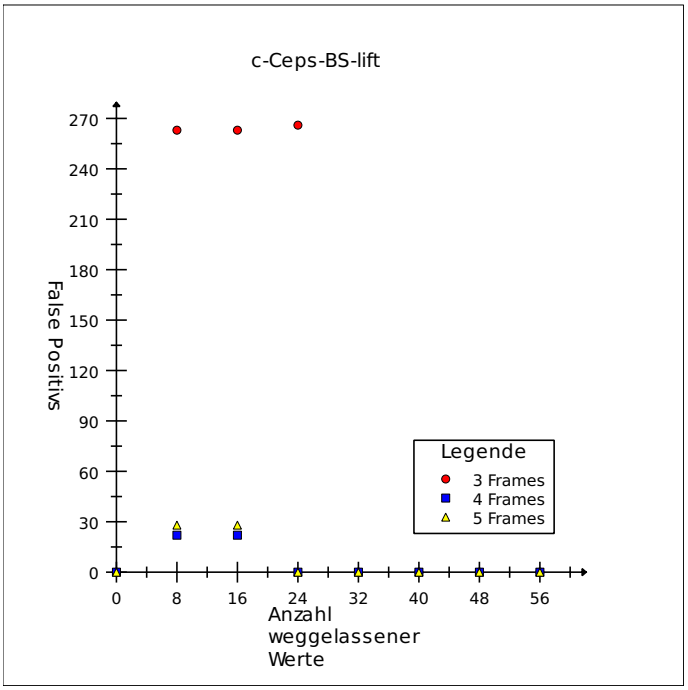

Abbildung 4.27: Cepstrum mit Liftering BS: Weggelassene Werte zu False Positivs

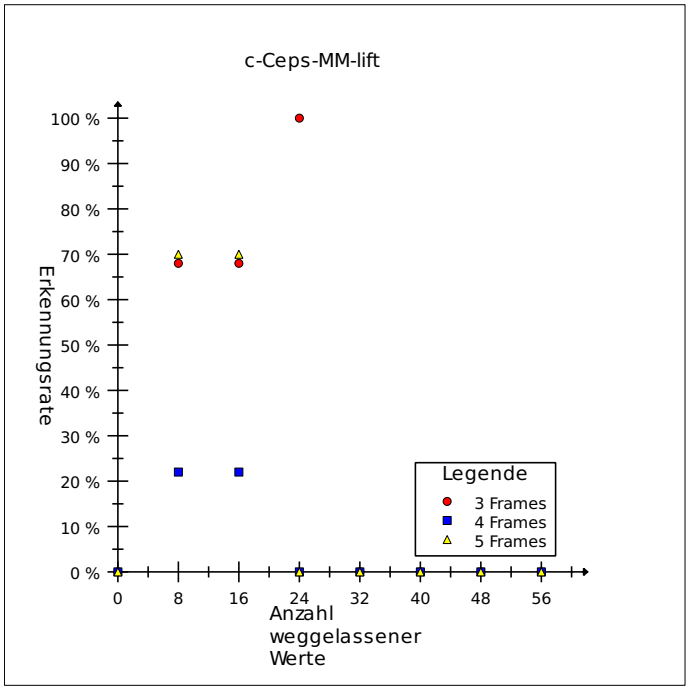

Abbildung 4.28: Cepstrum mit Liftering MM: Weggelassene Werte zu Erkennungsrate

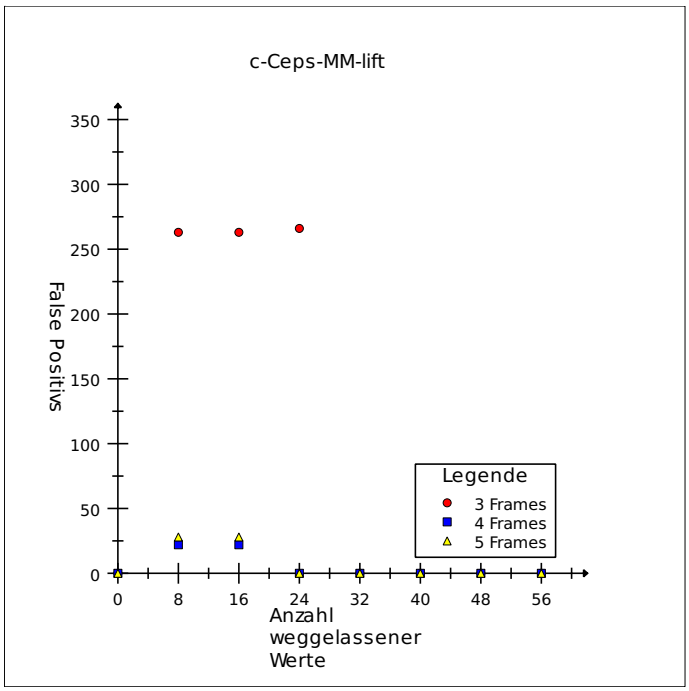

Abbildung 4.29: Cepstrum mit Liftering MM: Weggelassene Werte zu False Positivs

4.4.4 Modell D: pfeifen von zwei Tönen

Wie schon bei den beiden Verfahren zuvor, wurden Modells mit 3, 4 und 5 Frames und unterschiedlicher Anzahl an weggelassenen Werten berechnet.

4.4.4.1 Auswahlkriterium: Best Score

In Abbildung 4.30 ist zu sehen, dass es nur für keine, 8 und 16 weggelassene Werte überhaupt Ergebnisse gibt. Diese sehen sogar recht gut aus, allerdings fällt in Abbildung 4.31 auf, dass die Modells eine hohe Anzahl an False Positivs haben. Selbst das Modell mit den wenigsten False Positivs hat noch 113, also mehr als doppelt so viele wie es Testgeräusche gab.
Aufgrund der hohen False Positiv Rate wurde hier kein Modell ausgewählt.

4.4.4.2 Auswahlkriterium: Most Matches

Auch für dieses Auswahlkriterium sehen die Ergebnisse fast gleich aus wie für das Auswahlkriterium Best Score. In Abbildung 4.32 ist zu sehen, dass nur für 4 und 5 Frames und jeweils keine weggelassenen Werte die Erkennungsrate höher ist. In Abbildung 4.33 ist zu bemerken, dass diese beiden Modells dann logischerweise auch eine höhere Anzahl an False Positivs hat.
Auch hier wurden, genau wie oben, aufgrund der vielen False Positivs keine Modells ausgewählt.

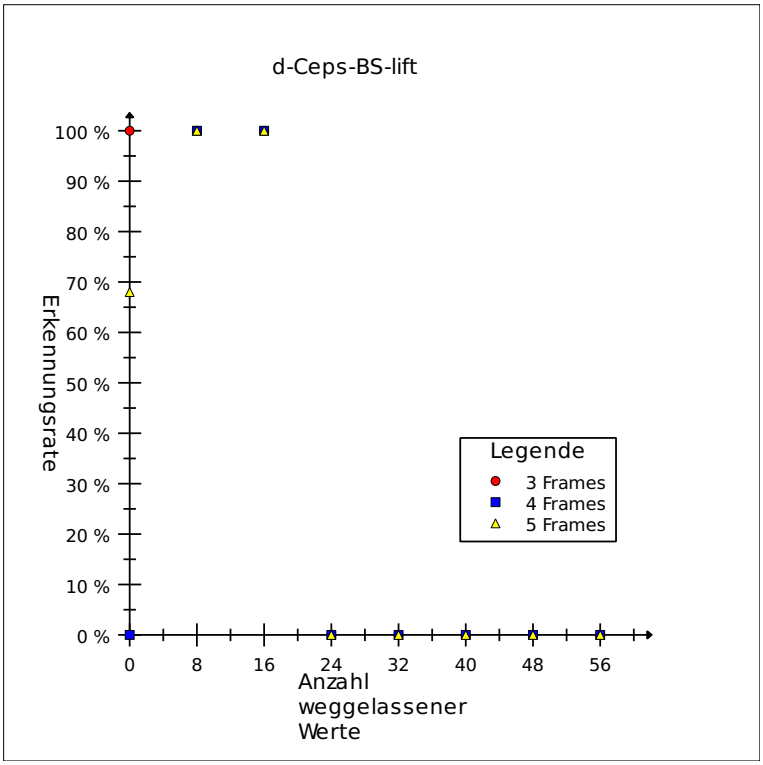

Abbildung 4.30: Cepstrum mit Liftering BS: Weggelassene Werte zu Erkennungsrate

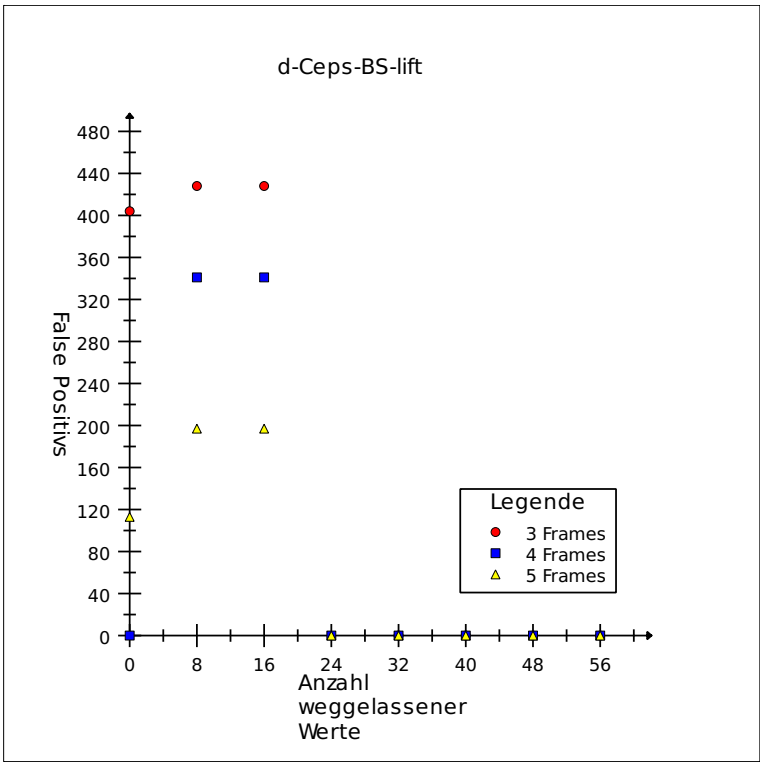

Abbildung 4.31: Cepstrum mit Liftering BS: Weggelassene Werte zu False Positivs

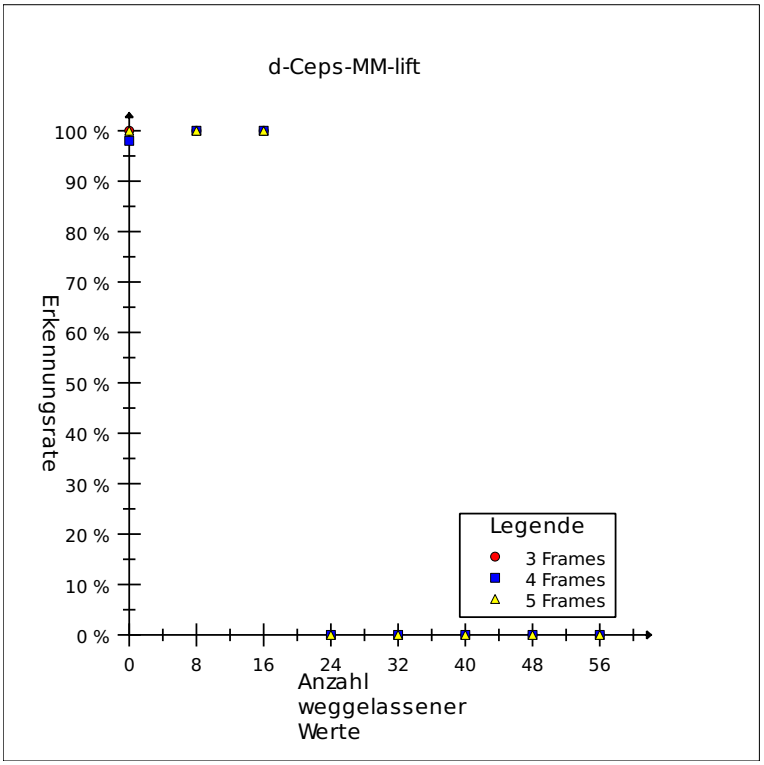

Abbildung 4.32: Cepstrum mit Liftering MM: Weggelassene Werte zu Erkennungsrate

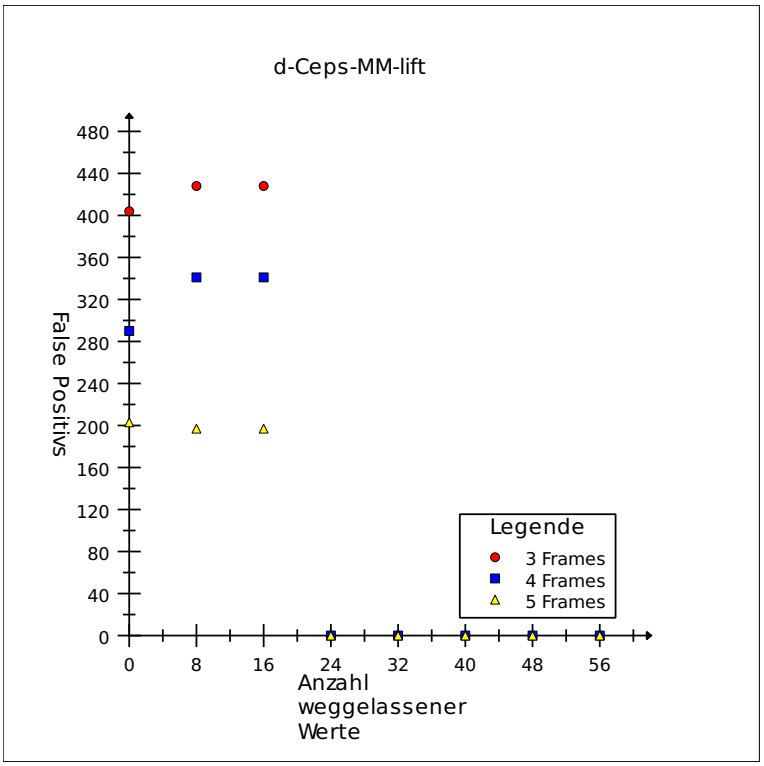

Abbildung 4.33: Cepstrum mit Liftering MM: Weggelassene Werte zu False Positivs

4.4.5 Modell E: Türklingel

Wie schon bei den beiden Verfahren zuvor, wurden Modells mit 3, 4 und 5 Frames und unterschiedlicher Anzahl an weggelassenen Werten berechnet.

4.4.5.1 Auswahlkriterium: Best Score

In Abbildung 4.34 ist zu sehen, dass es für die Modells mit 4 und 5 Frames ab 24 oder mehr weggelassenen Werten keine Ergebnisse mehr gibt. Insgesamt sind die Erkennungsraten sehr unterschiedlich verteilt. Wie in Abbildung 4.35 zu sehen ist, sind all diese Modells unbrauchbar, da sie alle mehr False Positivs als Testgeräusche hatten.
Aufgrund der hohen False Positiv Rate wurden hier keine Modells ausgewählt.

4.4.5.2 Auswahlkriterium: Most Matches

Wie in Abbildung 4.36 und Abbildung 4.37 zu sehen ist, sind die Ergebnisse ähnlich wie beim Auswahlkriterium Best Score und genau so unbrauchbar.
Auch hier wurden aufgrund der vielen False Positivs keine Modells ausgewählt.

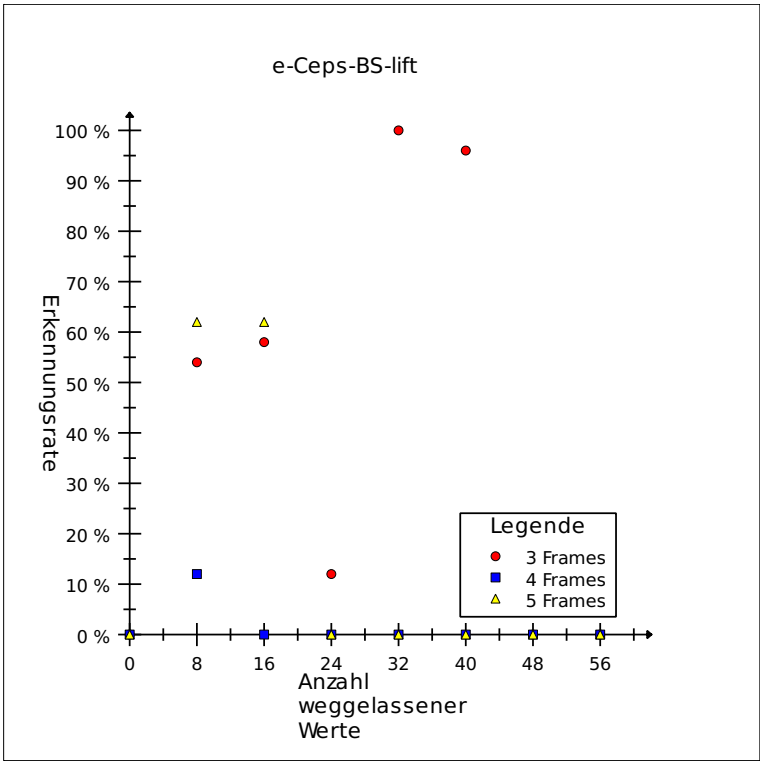

Abbildung 4.34: Cepstrum mit Liftering BS: Weggelassene Werte zu Erkennungsrate

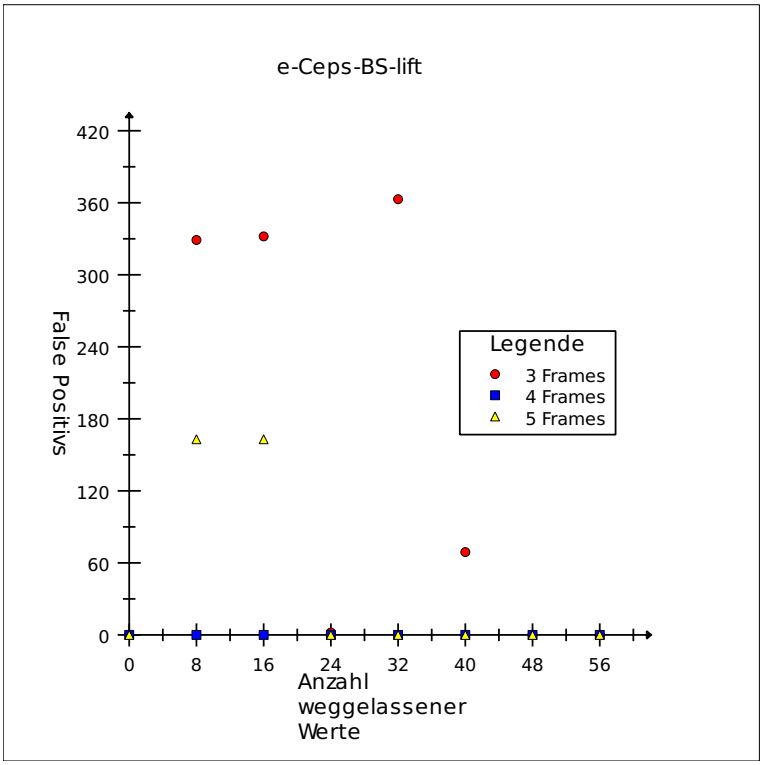

Abbildung 4.35: Cepstrum mit Liftering BS: Weggelassene Werte zu False Positivs

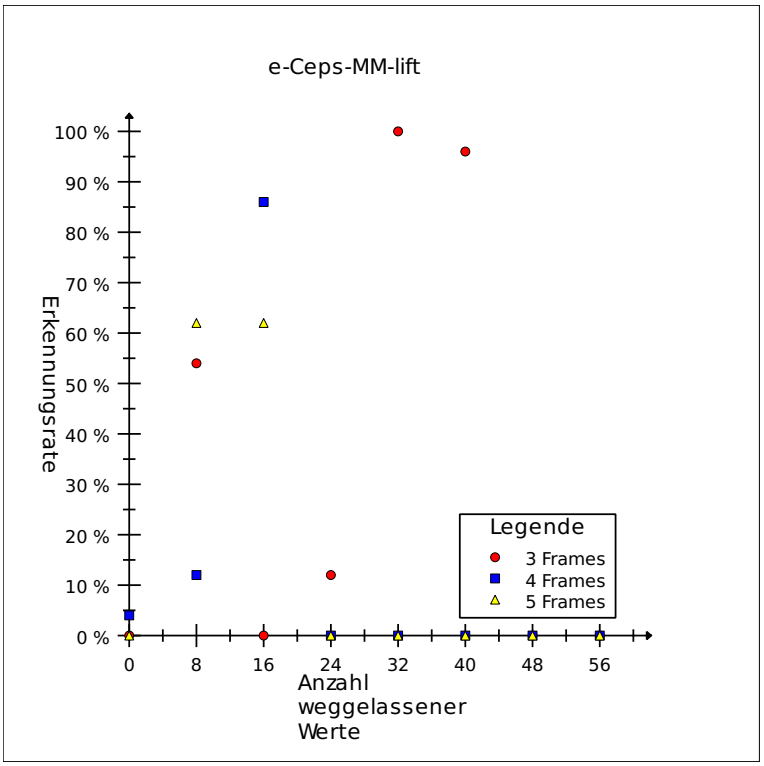

Abbildung 4.36: Cepstrum mit Liftering MM: Weggelassene Werte zu Erkennungsrate

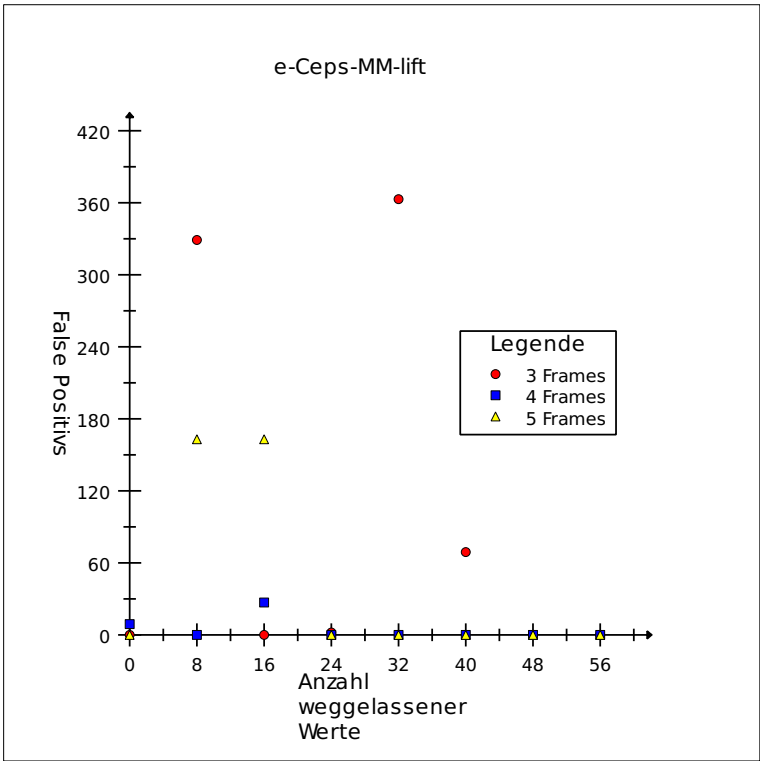

Abbildung 4.37: Cepstrum mit Liftering MM: Weggelassene Werte zu False Positivs

4.5 Tests mit Hintergrundgeräuschen

Da alle oben beschriebenen Experimente nur mit Testgeräuschen ohne Hintergrundgeräusche durchgeführt wurden, geben sie kein sonderlich realistisches Bild der Güte der hier beschriebenen Verfahren ab. In der realen Welt treten Geräusche normalerweise nicht isoliert und ohne Hintergrundgeräusche auf. Deswegen wurden die oben ausgewählten besseren Modells nochmals mit Hintergrundgeräuschen getestet. Dafür wurden auf Youtube Videos von Hintergrundgeräuschen gesucht, und deren Tonspur zusätzlich zu den Testgeräuschen abgespielt. Da pro Geräusch nur 50 Testaufnahmen gespeichert wurden, aber verschiedene Hintergrundgeräusche getestet werden sollten, wurden für jedes Hintergrundgeräusch alle 50 Testaufnahmen getestet. Insgesamt wurden 5 verschiedene Arten von Hintergrundgeräuschen genutzt, pro ausgewähltem Modell wurden also nochmal 250 Testgeräusche abgespielt.

Als Hintergrundgeräusche wurden folgende Videos ausgewählt:

* [1]: Lord of The Rings (Calm Ambient Mix): Soundtracks aus den Herr der Ringe Filmen.

* [6]: Harry Potter ASMR - Snape's potion classroom - Ambient Sound White Noise - potion boiling - HD: Eher dumpfe Geräusche von kochenden Flüssigkeiten sowie das Knistern eines Feuers.

* [3]: Harry Potter ASMR - Hogwarts Library - HD ambient sound white noise - Cinemagraphs: Eher höhere Geräusche, vor allem Regen und das Knistern von Feuer.

* [2]: One Hour of HQ Coffee Shop Background Noise: Hintergrundgeräusche in einem Coffeeshop. Vor allem Leute die Reden und Küchengeräusche.

* [8]: (3D binaural recording) Asmr sounds of cooking: Küchengeräusche.

Alle Videos, mit Ausnahme von [8], sind zwischen 50 und 65 Minuten lang. Da die Testgeräusche immer einen Abstand von einer Minute haben, sind die Testsets auch ca. 50 Minuten lang. [8] ist nur 25 Minuten lang und wurde deshalb zweimal hintereinander abgespielt. Somit wurden insgesamt ca. 5 Stunden Testgeräusche pro Modell getestet.

Die Beschriftung in den Legenden ist immer nach dem gleichen Schema aufgebaut. Zuerst kommt die Bezeichnung des Geräuschs durch einen Buchstaben. Dies sind genau die gleichen Buchstaben wie oben, a ist z.B. ein einfacher Pfeifton. Danach kommt die Anzahl der Frames, 3F bedeutet also, das ein Modell 3 Frames hat. Der dritte Block gibt an, wieviele Werte weggelassen wurden, z.B. 40 weggelassene Werte bei 40WW und zum Schluss steht das Auswahlkriterium, also BS für Best Score oder MM für Most Matches.

4.5.1 FFT mit lienarer Fensterung

In Abbildung 4.38 ist die Erkennungsrate für alle oben ausgewählten Modells zu sehen. Verglichen mit den Anzahlen der False Positivs, die in Abbildung 4.39 dargestellt sind, ist zu sehen, dass eine höhere Erkennungsrate meistens mit einer höheren Anzahl an False Positivs verbunden ist. Einzige Ausnahmen hierbei sind die Modells 6: b-5F-48WW-BS und 12: d-4F-40WW-MM, die keine False Positivs haben, obwohl sie beide eine Erkennungsrate von über 80 % erreichen.

Dieses Verfahren zur Merkmalsextraktion eignet sich also recht gut für die Geräusche b und d, also für das Schließen einer Tür und das Pfeifen von zwei Tönen. Die besten Modells dafür waren einmal das Modell b-5F-48WW-BS mit einer Erkennungsrate von 80,4 % und keinen False Positivs und das Modell d-4F-40WW-MM mit einer Erkennungsrate von 82,8 % und ebenso keinen False Positivs.
Für das Geräusch a, also das Pfeifen eines Tones eignet sich das Verfahren auch noch einigermaßen. Hier ist das beste Modell das Modell a-4F-40WW-BS mit einer Erkennungsrate von 66 % und 2 False Positivs.
Für das Geräusch c, also den Ofentimer, war dieses Verfahren nicht so gut geeignet. Das einzige Modell, das in den oben beschriebenen Experimenten ausgewählt wurde, war das Modell c-3F-0WW-BS, das eine Erkennungsrate von 54 % bei 73 False Positivs hat.
Für das Geräusch e, also die Türklingel, war dieses Verfahren nicht geeignet, es wurden schon gar keine Modells ausgewählt, um diese nochmals zu testen.

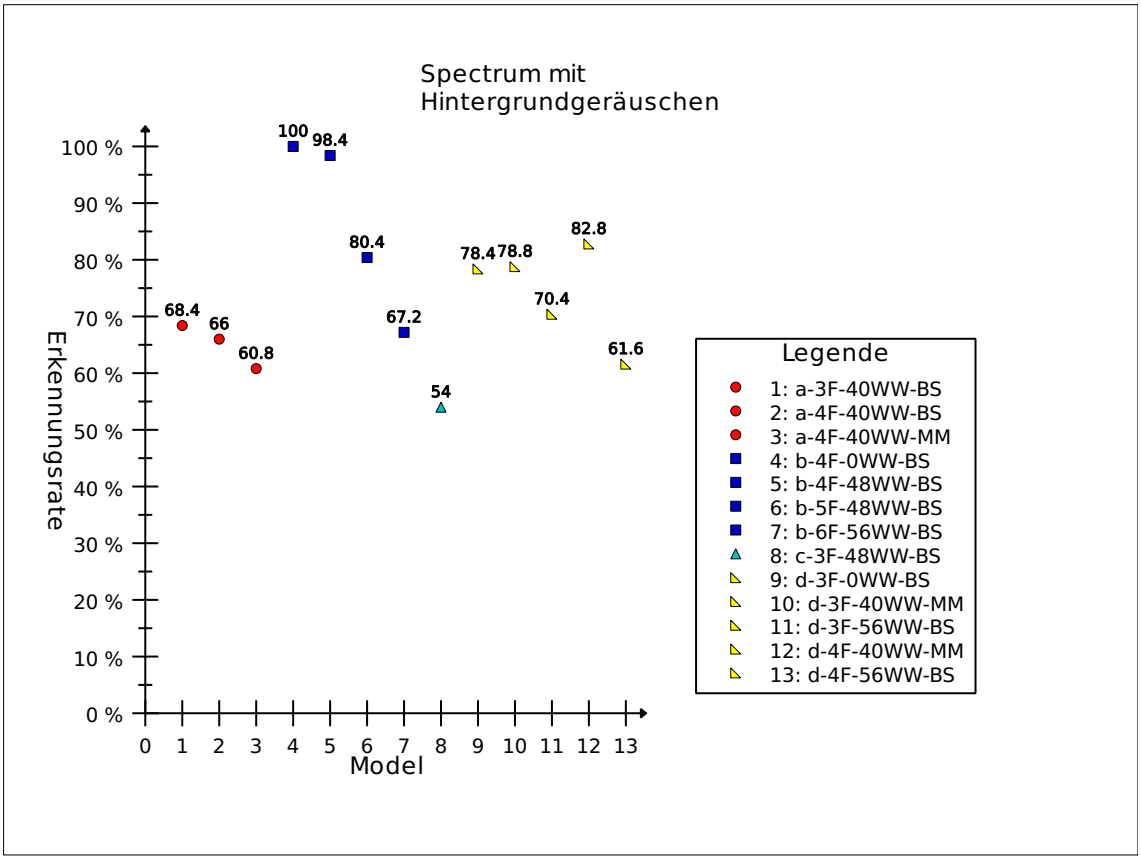

Abbildung 4.38: Spectrum mit Hintergrundgeräuschen: Erkennungsrate

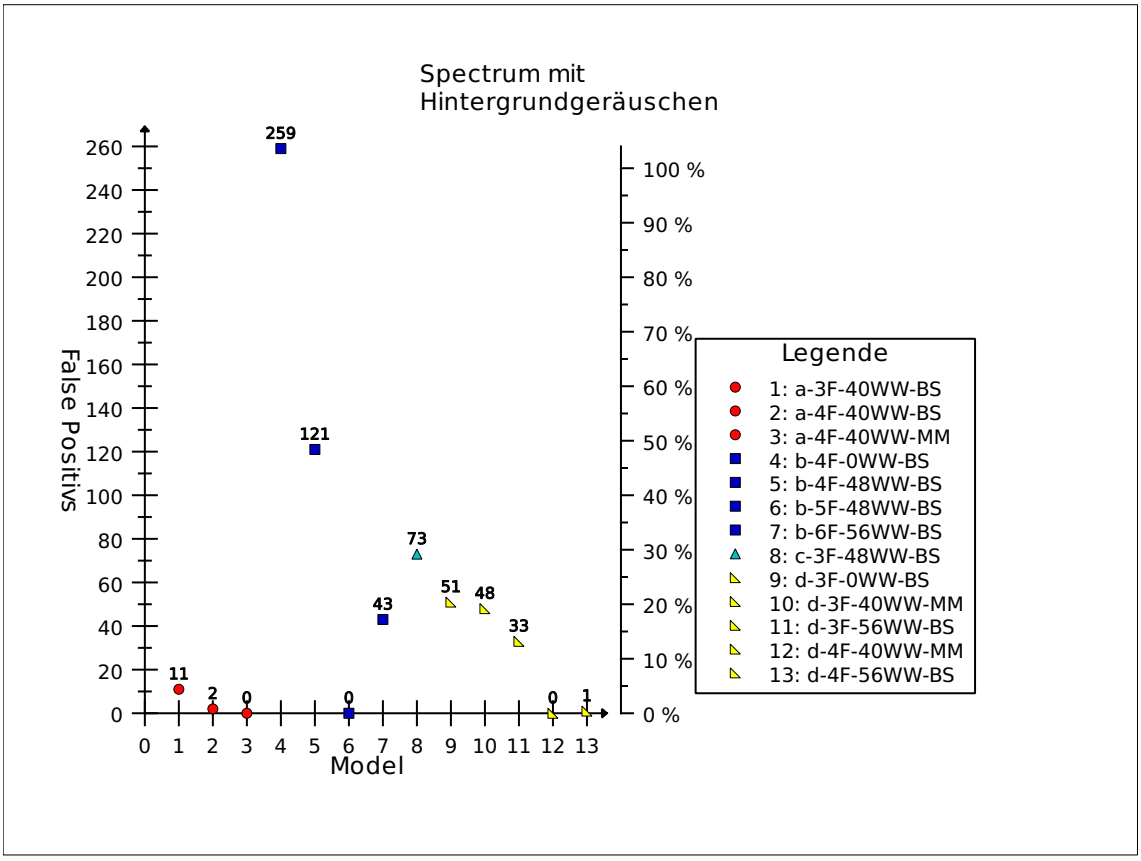

Abbildung 4.39: Spectrum mit Hintergrundgeräuschen: False Positivs

4.5.2 Cepstrum mit lienarer Fensterung ohne Liftering

Wird nur Abbildung 4.40 betrachtet, so erscheint es, als wäre dieses Verfahren leicht schlechter, als das Verfahren FFT. In Abbildung 4.41 fällt dann allerdings auf, dass dieses Verfahren wesentlich mehr False Positivs produziert. Lediglich die Modells 5, 6 und 7 haben eine Anzahl an False Positivs die unter Umständen noch als akzeptabel bezeichnet werden kann. Allerdings haben diese Modells auch eine, teilweise deutlich, schlechtere Erkennungsrate.

Dieses Verfahren zur Merkmalsextraktion eignet sich nur bedingt für Geräusche. Es sollte nur verwendet werden, wenn das vorherige Verfahren kein brauchbares Modell liefert.

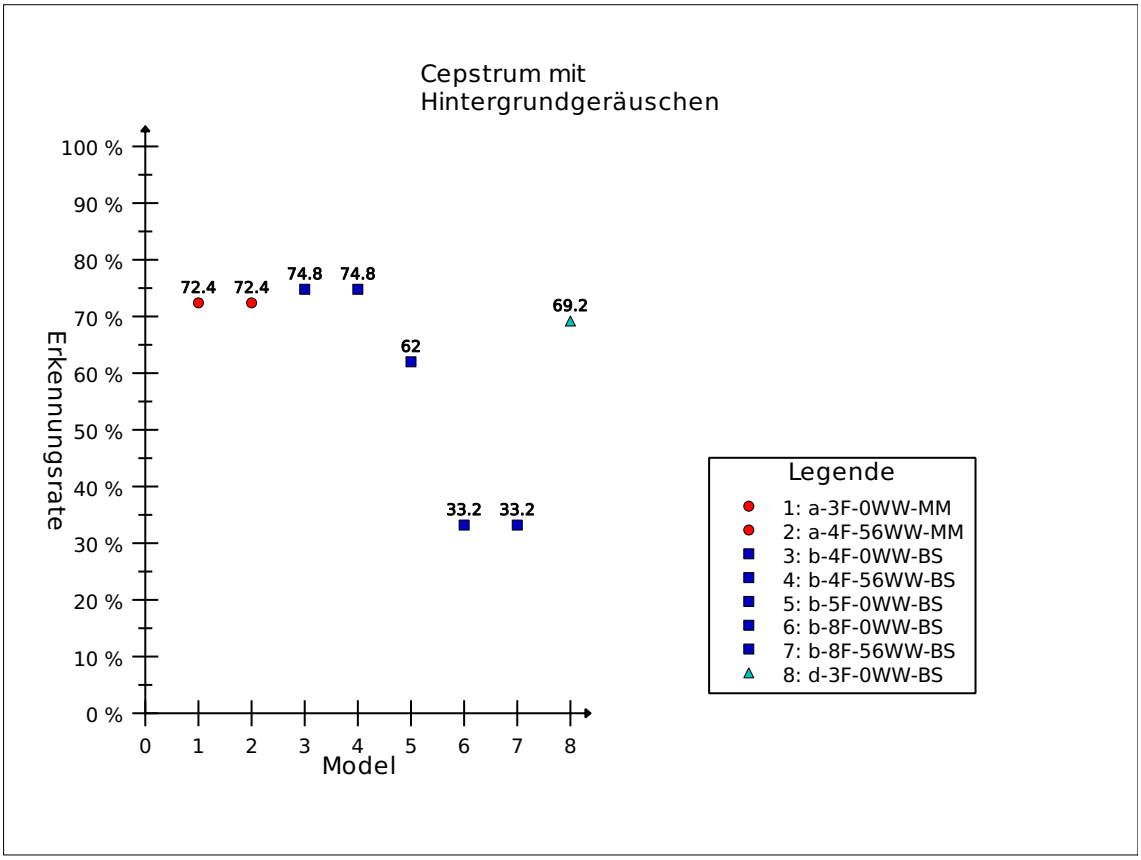

Abbildung 4.40: Cepstrum mit Hintergrundgeräuschen: Erkennungsrate

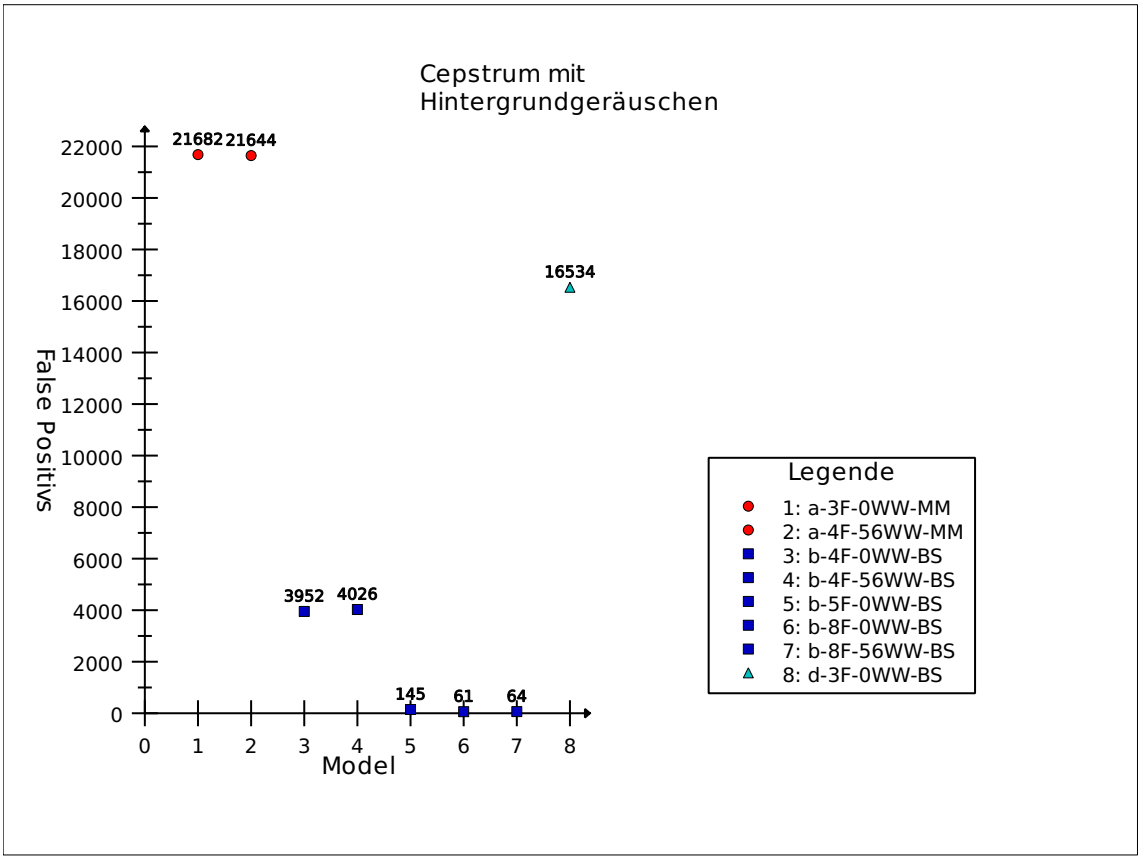

Abbildung 4.41: Cepstrum mit Hintergrundgeräuschen: False Positivs

4.5.3 Cepstrum mit lienarer Fensterung mit Liftering

Für dieses Verfahren wurden bereits in den vorhergegangenen allgemeinen Experimenten nur zwei Modells ausgewählt, um sie nochmal ausgiebig zu testen. Diese waren für das Geräusch c, also den Ofentimer. Beide hatten 5 Frames, und beim einen wurden 8 Werte weggelassen, beim anderen 16. Beide lieferten im Test die gleichen Ergebnisse, nämlich eine Erkennungsrate von 41,6 % und 626 False Positivs. Dies sind keine guten Ergebnisse, allerdings lieferte dieses Verfahren in den allgemeinen Experimenten für das Geräusch c wesentlich bessere Ergebnisse als das Verfahren Cepstrum ohne Liftering. Von daher könnte es für bestimmte Geräusche eventuell doch brauchbare Modells liefern, und sollte als Alternative ausprobiert werden, wenn die beiden oben genannten Verfahren kein nutzbares Modell liefern.

5 Diskussion und Ausblick

5.1 Diskussion

Insgesamt scheint es so, als wäre nur das Verfahren FFT mit linearer Filterung geeignet, um allgemeine Geräusche zu erkennen. Die Verfahren Cepstrum mit oder ohne Liftering eignen sich nur bedingt. Diese Verfahren werden vor allem in der Spracherkennung eingesetzt. Allerdings werden bei der Spracherkennung mehr Informationen als nur die Audiodaten verwendet. Unter anderem kommen dort ja auch Wörterbücher und grammatikalische Regeln zum Einsatz. Da hier allerdings nur Audiodaten betrachtet werden, funktionieren diese Verfahren nur mäßig.
Dies ist jedoch nicht verwunderlich, da das Cepstrum ja berechnet wurde, weil sich damit das Quelle-Filter Modell, von dem man bei der Sprachproduktion ausgeht, gut abbilden lässt. Bei den hier getesteten Geräuschen kann man allerdings nicht von einem solchen Quelle-Filter Modell ausgehen, weswegen dieses Verfahren auch nicht den Vorteil wie in der Spracherkennung bringt.

Wie bei jedem größeren Projekt, hatten sich auch bei diesem Projekt einige Programmierfehler eingeschlichen. So wurden zum Beispiel zwischendurch, beim Reduzieren der Werte innerhalb eines Frames von 512 auf 64, aufgrund eines kleinen Fehlers, nicht immer 8 Werte zusammengefasst, sondern alle Werte auf die ersten 8 Stellen des Merkmalsvektors verteilt. Eigentlich sollte beim Zusammenfassen der Index i jeden Wertes aus dem großen Array durch 8 geteilt werden, und der somit berechnete Index $j = i/8$ als Position im reduzierten Array genutzt werden. Im reduzierten Array werden die Werte, die an die gleiche Position geschrieben werden sollen, einfach aufaddiert. Der Fehler, der eine ganze Weile lang nicht auffiel, war, dass der neue Index j nicht durch $j = i/8$ sondern durch $j = i \bmod 8$ berechnet wurde, wodurch die Werte nicht gleichmäßig zusammengefasst, sondern auf die ersten 8 Stellen des Arrays verteilt wurden. Dies fiel allerdings erstmal nicht auf, da der Code trotzdem funktionierte.

Auch bei der Merkmalsextraktion schlichen sich ein paar Fehler ein, die unter anderem dazu führten, das einige Werte in den Arrays NaN wurden. Überraschenderweise funktionierte der Code trotzdem und lieferte sogar gute Ergebnisse. Daraus kann gefolgert werden, dass es gar nicht so wichtig ist, wie genau die Merkmale extrahiert werden, solange die Daten so weit reduziert werden, dass damit gearbeitet werden kann, ohne dass die Vorverarbeitung länger dauert als die Dauer des Frames.

Im Vergleich zu anderen Projekten wie beispielsweise die in Kapitel 1 kurz vorgestellten, sind die hier erzielten Ergebnisse durchaus vorzeigbar. Wird nur das Verfahren FFT mit

linearer Fensterung, das hier die besten Ergebnisse erzielt hat, betrachtet, so erreichen die ausgiebig getesteten Modells eine durchschnittliche Erkennungsrate von 74,4 %. Diese ist zwar etwas niedriger als das Projekt [7], das ICAs benutzt und auf eine Erkennungsrate von 83,4, aber höher als beim CHIL Projekt [5], das auf eine Erkennungsrate von 61,59 % kommt.

5.2 Ausblick

Wie sich im Laufe der Entwicklung zeigt, war Python wahrscheinlich die falsche Wahl, um einen Geräuscherkenner zu implementieren. Es wäre vermutlich besser gewesen, von vornherein in C zu programmieren, da damit zeitkritische Berechnungen performanter getätigt werden können. Dies fiel schon während der Implementierung auf, allerdings war zu diesem Zeitpunkt schon recht viel Code vorhanden, so dass eine Ńeuimplementierung verworfen wurde. Stattdessen wurde auf Cython gesetzt, eine weitgehend mit Python kompatible Programmiersprache, die in C übersetzt wird. Hiermit konnte auch schon der vorhandene Python Code mit leichten Modifikationen in C-Code übersetzt werden.

Ein weiterer Vorteil von C ist, dass Multithreading mit shared memory wesentlich besser unterstützt und somit wesentlich einfacher zu implementieren ist. Somit könnte vermutlich auch die Echtzeiterkennung vom Multithreading profitieren und es könnten rechenaufwändigere Verfahren zur Merkmalsextraktion sowie zur Entscheidung verwendet werden.

Literatur

[1] In: (). URL: http://www.orelia.fr/en/technology/audiosense-to-go-beyond-the-mere-sound-level.html.

[2] "Harry Potter ASMR - Hogwarts Library - HD ambient sound white noise - Cinemagraphs". In: (). URL: https://www.youtube.com/watch?v=20XE6GM7xWo.

[3] "Harry Potter ASMR - Snape's potion classroom - Ambient Sound White Noise - potion boiling - HD". In: (). URL: https://www.youtube.com/watch?v=eyYB-txU6Jg.

[4] Xuedong Huang, Alex Acero und Hsiao-Wuen Hon. *Spoken language processing : a guide to theory, algorithm, and system development.* Includes bibliographical references and index; Geb. : DM 182.95. Upper Saddle River, NJ: Prentice Hall, 2001. ISBN: 0-13-022616-5.

[5] Florian Kraft u. a. "Temporal ICA for Classification of Acoustic Events in a Kitchen Environment". In: (2013). URL: http://isl.anthropomatik.kit.edu/cmu-kit/667_Temporal_ICA_-_acoustic_events.pdf.

[6] "Lord of The Rings (Calm Ambient Mix)". In: (). URL: https://www.youtube.com/watch?v=UrJTZZibdHk.

[7] Robert Malkin u. a. "First evaluation of acoustic event classification systems in CHIL project". In: (2013). URL: http://isl.anthropomatik.kit.edu/cmu-kit/downloads/First_Evaluation_of_Acoustic_Event_Classification_Systems_in_the_CHIL_project.pdf.

[8] "One Hour of HQ Coffee Shop Background Noise". In: (). URL: https://www.youtube.com/watch?v=BOdLmxy06H0.