

**Semesterarbeit:**

# **SPEECH DETECTION**

**Florian Meier  
Alexander Schläpfer**

**1. März 2002**

# Inhaltsverzeichnis

<b>1</b>	<b>Aufgabenstellung</b> .....	<b>4</b>
1.1	Sprachdetektion.....	4
1.2	Aufgabe.....	4
1.3	Bericht.....	5
1.4	Termine, Bedingungen.....	5
<b>2</b>	<b>Einleitung</b> .....	<b>6</b>
<b>3</b>	<b>Autokorrelation</b> .....	<b>7</b>
3.1	Berechnung .....	7
3.2	Berechnung der Pitchfrequenz .....	8
3.3	Rauschsignal .....	9
<b>4</b>	<b>Optimierungsmethoden</b> .....	<b>11</b>
4.1	Digitaler Filter .....	11
4.2	Center Clipping.....	12
4.3	Fazit.....	16
<b>5</b>	<b>Signalleistung</b> .....	<b>17</b>
5.1	Berechnung .....	17
5.2	Sprachsignal.....	20
5.3	Geräusch.....	20
5.4	Längere Signale .....	22
5.5	Rauschen überlagern .....	23
<b>6</b>	<b>Detektion</b> .....	<b>25</b>
6.1	Variante 1 .....	25
6.2	Variante 2.....	27
<b>7</b>	<b>Beispiele</b> .....	<b>29</b>
7.1	Sprachsignale.....	30
7.2	Geräusche.....	39
7.3	Gemischte Signale.....	48
<b>8</b>	<b>Erweiterungsmöglichkeit</b> .....	<b>56</b>
<b>9</b>	<b>Schlusswort</b> .....	<b>57</b>
<b>10</b>	<b>Literaturverzeichnis</b> .....	<b>58</b>
<b>A</b>	<b>Listing Matlab</b> .....	<b>59</b>
A.1	p_track.m .....	59
A.2	center_clipping1.m (Variante Amax).....	60
A.3	center_clipping.m (Variante Hüllkurve).....	61

A.4	leist.m .....	61
A.5	ausgabe.m.....	62
A.6	signalrausch.m.....	63
<b>B</b>	<b>Listing C++ .....</b>	<b>64</b>
B.1	header.h .....	64
B.2	Operationen .....	65
B.2.1	OpenWaveFile.....	65
B.2.2	CharToFloat.....	65
B.2.3	AutoCorr .....	66
B.2.4	Power .....	67
B.2.5	Detection .....	67
B.2.6	Filter .....	68
B.2.7	GetKoeff .....	69
B.2.8	Dialog .....	70
B.2.9	WriteTxtFile.....	71

# 1 Aufgabenstellung

## 1.1 Sprachdetektion

Das Ziel der Sprachdetektion ist es, in einem digitalen akustischen Signal festzustellen, wann Sprache vorhanden ist und wann nicht. Dazu wird das Signal auf typische Merkmale gesprochener Sprache hin untersucht: stimmhafte Signalabschnitte, zeitliche Verteilung der Signalleistung, zeitlicher Verlauf der spektralen Envelope usw.

Mit der stetig zunehmenden Leistungsfähigkeit digitaler integrierter Schaltungen wird die Sprachdetektion auch für Anwendungen in Hörgeräten interessant. Dabei geht es darum, gesprochene Sprache sowohl in ruhiger Umgebung als auch im Störlärm zu erkennen, wie auch Sprache von Musik zu unterscheiden und damit schliesslich die Verarbeitung des akustischen Signals geeignet zu beeinflussen.

Die Grundlage zum Erkennen stimmhafter Signalabschnitte bilden verschiedene Ansätze zur Bestimmung der Pitchperiode. Das Ziel ist jeweils, den periodischen Signalverlauf zu erkennen, welcher durch die periodische Aktivität der Stimmbänder zu Stande kommt.

Gesprochene Sprache zeichnet sich ebenfalls durch eine besondere zeitliche Verteilung der Signalleistung aus, welche durch den typischen Silbenrhythmus zu Stande kommt. Eine Analyse ist sowohl über ein breites Frequenzband als auch in vielen schmalen Teilbändern möglich.

Schliesslich ist auch der zeitliche Verlauf der spektralen Envelope typisch, durch welche die einzelnen aufeinander folgenden Phoneme charakterisiert sind. Sie werden sinnvollerweise mit Kepstralkoeffizienten erfasst.

## 1.2 Aufgabe

Die Studienarbeit lässt sich in folgende Teilaufgaben gliedern:

- Kennenlernen der Methoden zur Schätzung der Pitchperiode.
- Entwickeln eines Verfahrens zur Sprachdetektion auf der Grundlage der Analyse stimmhafter Signalabschnitte.
- Entwickeln eines Verfahrens zur Sprachdetektion auf der Grundlage der Analyse der zeitlichen Verteilung der Signalleistung.
- Kennenlernen der Methoden zur Bestimmung spektraler Envelopen.
- Entwickeln eines Verfahrens zur Sprachdetektion auf der Grundlage der Analyse des zeitlichen Verlaufs der spektralen Envelope.
- Entwickeln eines Verfahrens zur Sprachdetektion durch die Kombination der Analyse hinsichtlich verschiedener Sprachmerkmale.
- Entwickeln der Algorithmen zunächst in Matlab, anschliessend in einer höheren Programmiersprache (C/C++ oder Java) mit Festkomma-Arithmetik.
- Erproben der verschiedenen Verfahren mit umfangreichen Signalbeispielen.
- Dokumentation mit Schwergewicht auf den selbst geleisteten Arbeiten.

### 1.3 Bericht

Über die Arbeit ist ein Bericht zu verfassen. Dabei sind die Richtlinien der Abteilung für Elektrotechnik für das Erstellen von Berichten einzuhalten. Alle verwendeten Quellen sind im Literaturverzeichnis des Berichts anzugeben (Hinweis im Text). Der Bericht ist in doppelter Ausführung abzugeben; ein Exemplar verbleibt im Labor für digitale Signalverarbeitung der HSR, das Doppel erhalten die Verfasser nach der Korrektur und der Bewertung zurück. Die erstellten Programme und der Text des Berichtes sind auf Diskette(n) (mit der Nummer der Studienarbeit) beizulegen.

### 1.4 Termine, Bedingungen

Ausgabe der Aufgabenstellung: 13. November 2001

Abgabe des Zwischenberichts: 21. Dezember 2001

Abgabe des Berichts, Ende der Arbeit: 1. März 2002

### Kontaktadresse

Bernafon AG  
A. Schaub  
Eichtalstr. 55  
8634 Hombrechtikon  
Tel. 055 - 264 13 51  
E-Mail: [as@bernafon.ch](mailto:as@bernafon.ch)

## 2 Einleitung

Die in der nachfolgenden Dokumentation beschriebenen Programme gehen immer davon aus, dass die verwendeten WAV-Dateien eine Abtastfrequenz  $f_{\text{abtast}}$  von 22050 Hz und eine Genauigkeit `BitsPerSample` von acht Bit haben sowie nur einen Kanal (MONO) aufweisen. Jede Detektionsart wurde zuerst in MATLAB implementiert. Um die Auswertgeschwindigkeit zu erhöhen, wurden zu einem späteren Zeitpunkt alle Programme zusätzlich in der Hochsprache C++ implementiert. Die Ausgabe geschieht weiterhin mit MATLAB. Dafür werden die benötigten Daten in ein Text-File geschrieben und danach mittels dem m-File `ausgabe.m` geplottet.

## 3 Autokorrelation

### 3.1 Berechnung

Eine sehr verbreitete Methode zur Bestimmung der Pitchperiode ist die Autokorrelation. Da die Grundfrequenz der Sprache innerhalb von ca. 80 bis 600 Hz liegt muss man nur bei einer Verschiebung von  $T_{\min}$  bis  $T_{\max}$  die Autokorrelation berechnen.

$$\begin{aligned}
 T_{\min} &= \frac{1}{600\text{Hz}} \\
 T_{\max} &= \frac{1}{80\text{Hz}}
 \end{aligned}
 \tag{3.1}$$

Für die Berechnung der Autokorrelation wurde mittels Matlab ein m-File `p_track` geschrieben. Darin werden  $T_{\min}$  und  $T_{\max}$  in Samples ausgedrückt. Die Abtastfrequenz  $f_{\text{abtast}}$  der verwendeten Signale beträgt wie in der Einleitung erwähnt 22050 Hz.

$$\begin{aligned}
 T_{\min} &= \frac{f_{\text{abtast}}}{600\text{Hz}} = 36,75 \rightarrow 30 \text{ Samples} \\
 T_{\max} &= \frac{f_{\text{abtast}}}{80\text{Hz}} = 275,625 \rightarrow 300 \text{ Samples}
 \end{aligned}
 \tag{3.2}$$

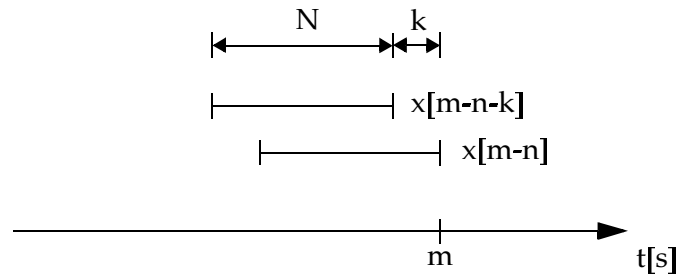
Die Berechnung erfolgt mit der nachfolgenden Formel:

$$\rho(m, k) = \frac{\sum_{n=0}^{N-1} x[m-n] \cdot x[m-n-k]}{\sqrt{\sum_{n=0}^{N-1} x[m-n]^2 \cdot \sum_{n=0}^{N-1} x[m-n-k]^2}}$$

$$\rho(m) = \max[\rho(m, k)]
 \tag{3.3}$$

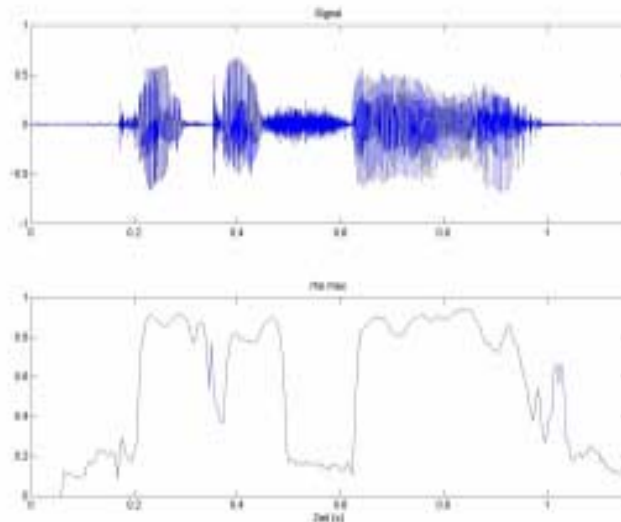
$$k \in [T_{\min}, T_{\max}]$$

$m$  = ganzzahlige „Zeit“-Variable  
 $k$  = ganzzahlige „Zeit“-Verschiebung  
 $n$  = Summationsvariable



**Abb. 3.1:** Verschiebung Autokorrelation

### 3.2 Berechnung der Pitchfrequenz



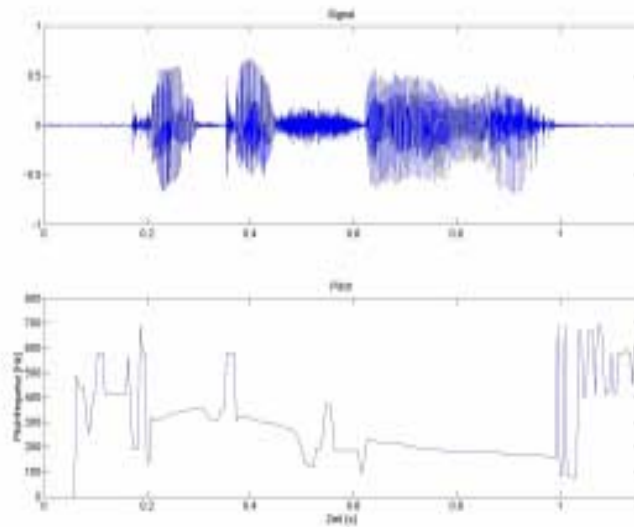
**Abb. 3.2:** Maximalwertberechnung der Autokorrelation

Die Autokorrelation ergibt Werte zwischen -1 und 1. Bei zwei sehr ähnlichen Signalabschnitten bzw. periodischen Signalen liegen die Werte der Autokorrelation annähernd bei 1. Da auch ein stimmhaftes Signal eine Periode (sog. Pitchperiode verursacht durch die Schwingung der Stimmbänder) aufweist, verschiebt man zwei Signalabschnitte übereinander, bis die Verschiebung der Pitchperiode entspricht. Dann sind die beiden Signalabschnitte ähnlich, das Signal korreliert sehr gut. Mit der Position dieses Maximums lässt sich die Pitchperiode  $P$  berechnen.

$$P = \frac{k|_{\rho_{max}}}{f_{abstast}} \quad 3.4$$

Daraus lässt sich die Pitchfrequenz  $f_{pitch}$  bestimmen (Abb. 3.3).

$$f_{pitch} = \frac{1}{P} \quad 3.5$$

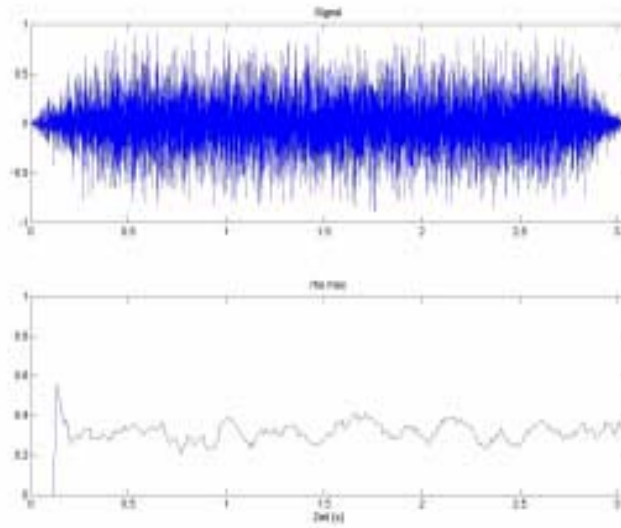


**Abb. 3.3:** Pitchfrequenz

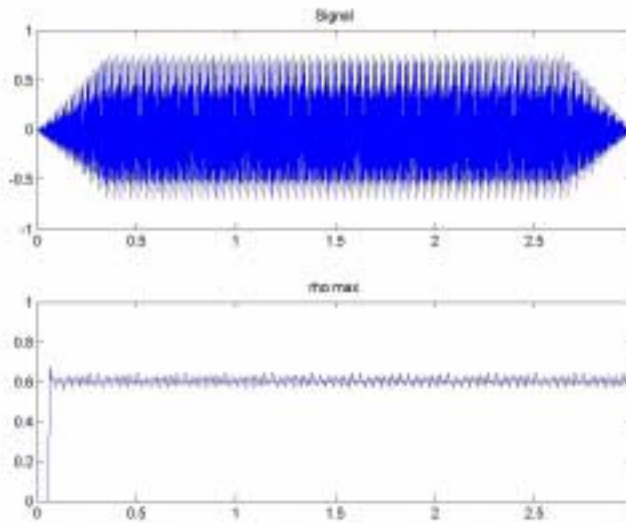
Weil periodische Signale (z.B. ein Musiksignal, Sinussignal) auch eine sehr hohe Autokorrelation aufweisen, benötigt man noch weitere Anhaltspunkte zur Bestimmung, ob es sich um ein stimmhaftes oder stimmloses Signal handelt (Siehe "Signalleistung" auf Seite 17).

### 3.3 Rauschsignal

Ein stochastisches Rauschsignal dagegen korreliert überhaupt nicht. Die Autokorrelationswerte liegen nicht annähernd bei 1 (Abb. 3.4 und Abb. 3.5 Seite 10).



**Abb. 3.4:** 1. Rauschsignal



**Abb. 3.5:** 2. Rauschsignal

## 4 Optimierungsmethoden

Um das Resultat der Autokorrelation zu optimieren, versucht man zusätzliche Methoden einzusetzen, um die Störeinflüsse (z.B Rauschen im Signal) zu eliminieren. Das Signal wird durch Filterung und Center Clipping vor der Autokorrelation bearbeitet, um eine gewünschte Verbesserung in der Sprachdetektion zu erreichen.

### 4.1 Digitaler Filter

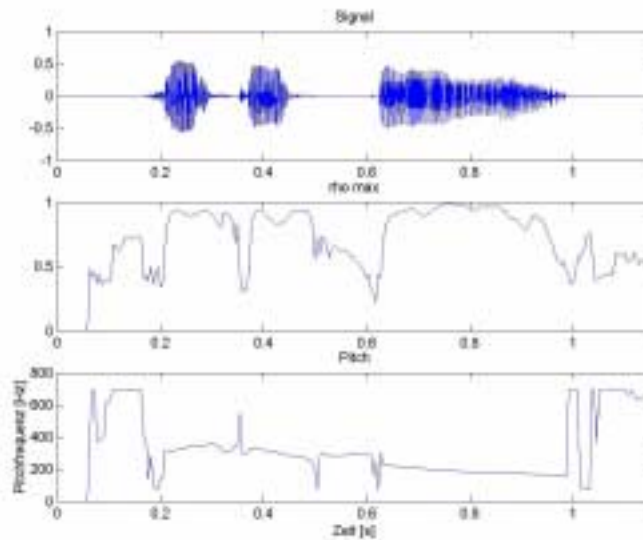
In der Sprache stecken nur Grundfrequenzen bis ca. 600 Hz. Die hohen Frequenzen werden mit einem digitalen Filter gedämpft. Es wurde ein Butterworth-Filter eingesetzt. Es genügt eine niedrige Filterordnung zur Erzielung guter Ergebnisse. Die Auslegung dieses IIR-Tiefpassfilter geschieht mit dem Befehl (MATLAB)

```
[B, A] = butter(ordnung, grenzfrequenz)
```

der die Koeffizienten-Vektoren der Übertragungsfunktion zurückgibt. Die normierte Grenzfrequenz muss im Bereich 0..1 liegen, wobei der Wert 1 der halben Abtastfrequenz entspricht. Der Filter wird mit dem Befehl

```
output = filter(B, A, input)
```

auf das Signal angewandt.



**Abb. 4.1:** Sprachsignal gefiltert

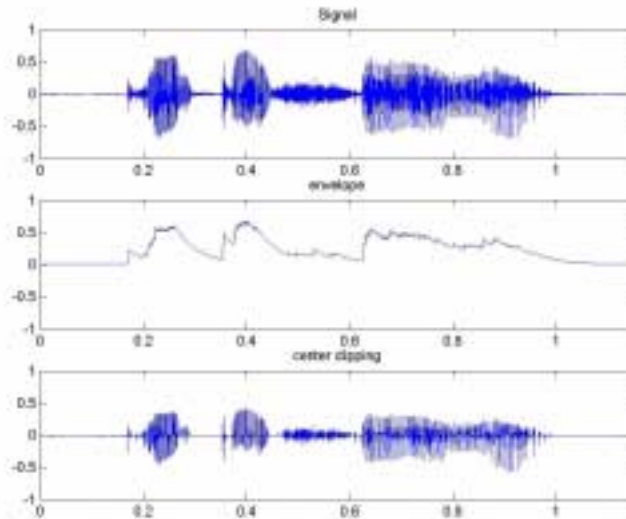
Wie aus Abb. 4.1 ersichtlich ist, konnte mit dieser Methode kein besseres Resultat erzielt werden. An den Stellen, wo keine Sprache herrscht, ist eine höhere Autokorrelation aufgetreten als bei Abb. 3.2 Seite 8. Bei Sprache hingegen sind die Resultate noch höher ausgefallen, was als ein positives Ergebnis zu werten ist. Trotzdem ist von dieser Methode abzuraten, da der Unterschied der Autokorrelationswerte zwischen stimmhaften und stimmlosen Abschnitten kleiner wird. Durch Dämpfung der hohen Frequenzen wird das Rauschen (stochastisch), das gerade den Unterschied zur Sprache ausmacht, entfernt. Der Nutzen dieser Methode ist fraglich.

## 4.2 Center Clipping

Bei dieser Methode werden Schwingungen mit geringer Amplitude entfernt. Es wurden zwei verschiedene Varianten angewendet.

Bei der ersten Variante, wird eine Hüllkurve  $h[n]$  um das Signal  $x[n]$  gezogen. Die Hüllkurve folgt hauptsächlich den Extremwerten (Abb. 4.2 Seite 13). Ist ein Maximum erreicht, fällt die Kurve langsam ab bis ein neuer Extremwert die Kurve überschreitet.

$$h[n] = \max(0, 9999 \cdot h[n-1], x[n]) \quad 4.1$$



**Abb. 4.2:** Center Clipping Var. 1

Das Ausschneideverfahren erfolgt bei beiden Varianten gleich (Formel 4.4). Bei der ersten Variante wird die gestauchte Hüllkurve dazu verwendet (Formel 4.2).

$$C[n] = 0,4 \cdot h[n] \quad 4.2$$

Bei der zweiten Variante wird das Maximum in einem Fenster detektiert (Formel 4.3). Anschliessend wird ein Band (Formel 4.4) aus dem Signal ausgeschnitten (Abb. 4.3 Seite 14).

$$C = 0.4 \cdot A_{max} \quad 4.3$$

$$y[n] = \begin{cases} x[n] - C & \text{für } x[n] \geq C \\ x[n] + C & \text{für } x[n] \leq -C \\ 0 & \text{sonst} \end{cases} \quad 4.4$$

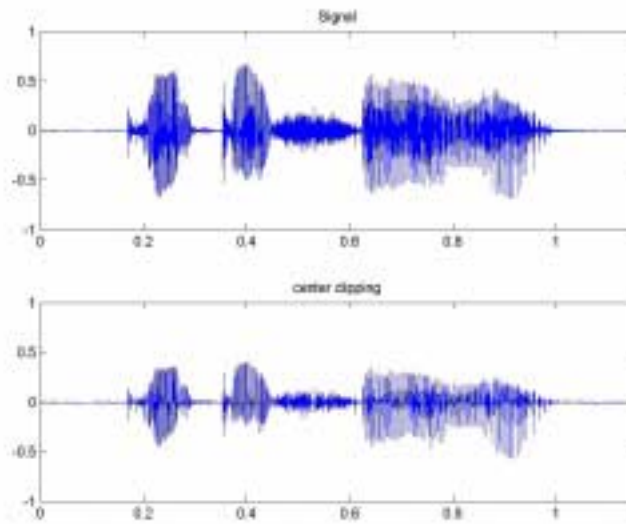


Abb. 4.3: Center Clipping Var. 2

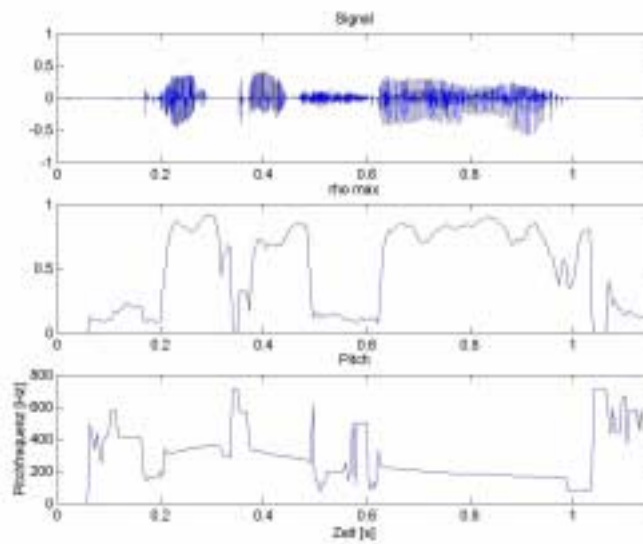
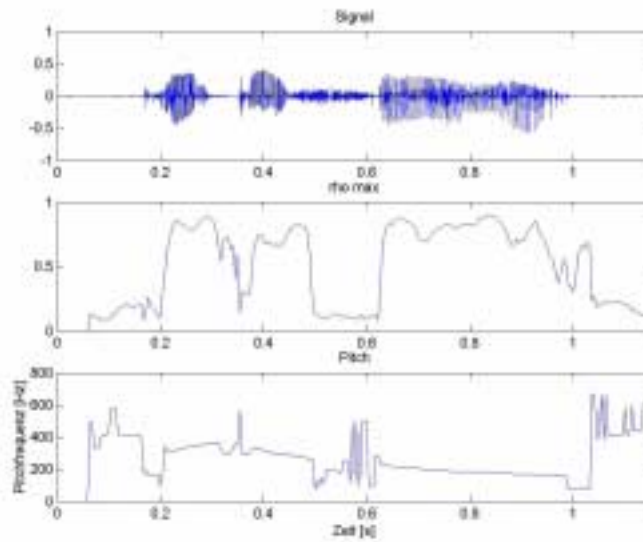


Abb. 4.4: Pitchfrequenz mit Center Clipping Var. 1



**Abb. 4.5:** *Pitchfrequenz mit Center Clipping Var. 2*

Diese Vorbereitung des Signals bringt keine wesentliche Verbesserung. Die beiden Varianten liefern ähnliche Resultate, wobei bei der ersten Variante (Abb. 4.4 Seite 14) gut zu erkennen ist, wie nach einem Sprachabschnitt die Autokorrelation rasch klein wird. Dies kann für ein weiteres Vorgehen der Detektion der Sprache hilfreich sein.

### 4.3 Fazit

Mit den zusätzlichen Methoden wurde nicht die gewünschte Verbesserung erzielt. Die Tiefpassfilterung ist wegen seinen unerwünschten Periodizitäten in den stimmlosen-Abschnitten nicht zu empfehlen.

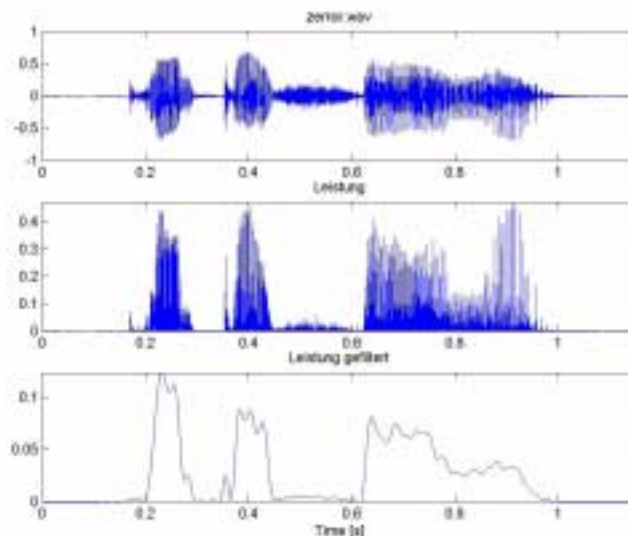
Das Center Clipping lässt sich nicht weiter optimieren. Es wurde in zwei verschiedenen Varianten untersucht. Im Gegensatz zur Tiefpassfilterung tritt hier keine Verschlechterung auf. Es wird sich zeigen, ob für das weitere Vorgehen der Sprachdetektion die Autokorrelation genügend ist. Zur Sprachdetektion müssen weitere Sprachmerkmale berücksichtigt werden.

## 5 Signalleistung

### 5.1 Berechnung

Ein weiteres Verfahren zur Sprachdetektion ist die zeitliche Verteilung der Signalleistung. Es wurde mittels MATLAB ein m-File `leist.m` implementiert. Zuerst berechnet man die Leistung des Signals  $x[n]$  für jedes Sample

$$P[n] = (x[n])^2 \quad 5.1$$



**Abb. 5.1:** *Leistung, gefiltert*

Danach wird die Leistung  $P[n]$  gefiltert (Abb. 5.1). Es wird wiederum ein Butterworthfilter mit der Ordnung 6 verwendet. Die Grenzfrequenz liegt bei 50 Hz.

$$[B \ A] = \text{butter}(6, 50\text{Hz}/(f_{\text{abtast}}/2))$$

Die Filterung erfolgt mit dem Befehl

$$\text{output} = \text{filter}(B, A, \text{input}).$$

Im nächsten Schritt bildet man eine obere und untere Envelope  $e_{\text{top}}$  bzw.  $e_{\text{bottom}}$ . Zu beachten ist, dass die Zeitkonstante  $T$  relativ gross gewählt wird.

$$T = 1s \quad 5.2$$

Dadurch sind die Envelopen erst nach ca.  $5T$  abgeklungen.

$$e_{top}[n] = \max(P[n], e_{top}[n-1] \cdot top) \quad 5.3$$

$$e_{bottom}[n] = \begin{cases} \min(P[n], e_{bottom}[n-1] \cdot bottom) & \text{für } P[n] > u_{Grenze} \\ u_{Grenze} & \text{sonst} \end{cases} \quad 5.4$$

Falls  $P[n]$  einmal gleich Null wird, würde  $e_{bottom}$  für immer Null bleiben. Damit dieser Fall nie eintritt, führt man eine untere Grenze  $u_{Grenze}$  ein. Sie entspricht dem kleinsten möglichen Wert der Leistung. Bei einem mit `BitsPerSample` gleich acht Bit abgetasteten Signal entspricht

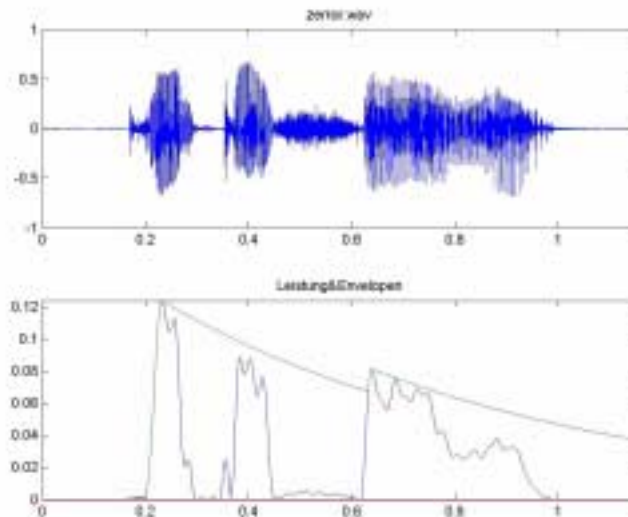
$$u_{Grenze} = \left( \frac{1}{2^{BitsPerSample-1}} \right)^2 \quad 5.5$$

Die obere Grenze  $o_{Grenze}$  ist eins, da das Signal nie grösser als eins ist und somit auch die Leistung diesen Betrag nie überschreitet.

Damit die Envelopen innerhalb von  $5T$  eingeschwungen sind, werden die Faktoren `top` bzw. `bottom` wie folgt berechnet,

$$top = \left( \frac{u_{Grenze}}{o_{Grenze}} \right)^{\frac{1}{5 \cdot T \cdot f_{abstast}}} \quad 5.6$$

$$bottom = \left( \frac{o_{Grenze}}{u_{Grenze}} \right)^{\frac{1}{5 \cdot T \cdot f_{abstast}}} \quad 5.7$$

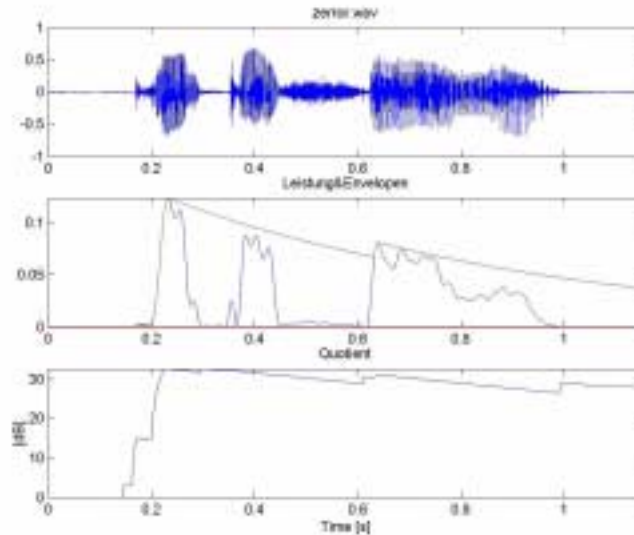


**Abb. 5.2:** Untere und Obere Envelope

Falls der Quotient  $Q$  von  $e_{top}$  und  $e_{bottom}$  grösser als ein bestimmter Schwellwert  $threshold$  ist, kann das Signal als ein Sprachsignal gedeutet werden.  $threshold$  wurde auf 30 gesetzt, was 15 dB entspricht. Da sehr grosse Zahlen möglich sind (maximal  $o_{grenze}/u_{grenze} = 16384$ ), stellt man den Quotienten in dB dar. Wenn dieser Quotient grösser als 15 dB ist, kann man sagen, dass es sich um ein Sprachsignal handelt (Abb. 5.3).

$$Q[n] = \frac{e_{top}[n]}{e_{bottom}[n]} > threshold \rightarrow \text{Sprachsignal} \quad 5.8$$

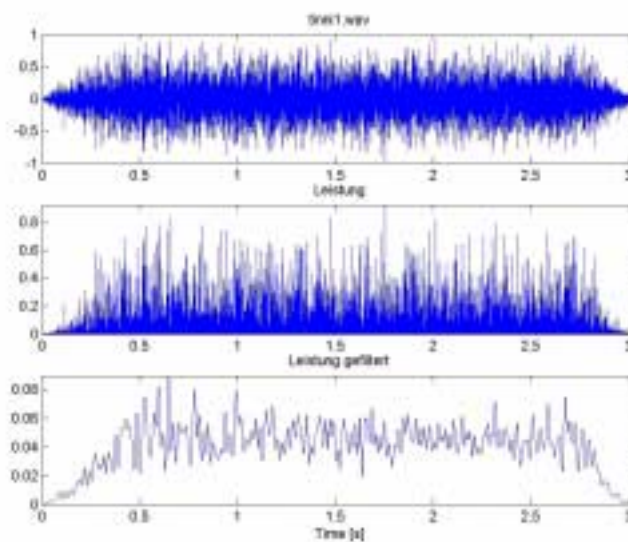
## 5.2 Sprachsignal



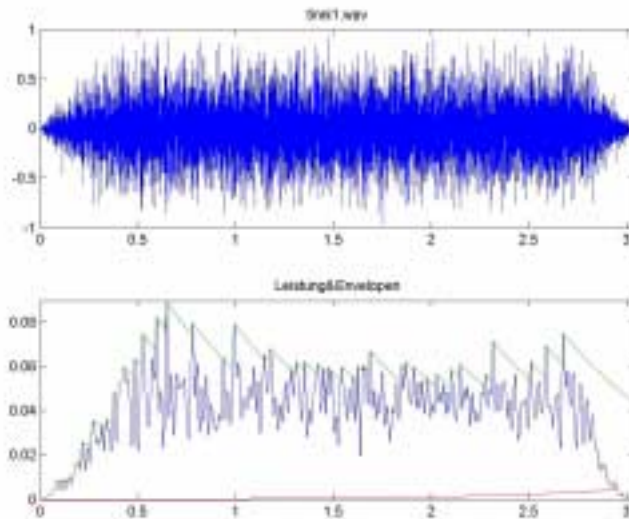
**Abb. 5.3:** Quotient Sprachsignal

Wenn man ein Geräusch auswertet, sieht man, dass der Quotient in Abb. 5.6 Seite 21 die ersten paar Sekunden auch grösser ist wie `threshold`. Dies weil die Envelopen erst nach  $5T$  eingeschwungen sind (Abb. 5.5 Seite 21). Deshalb darf man mit dieser Methode in den ersten paar Sekunden nicht entscheiden, ob es sich um ein Sprachsignal handelt.

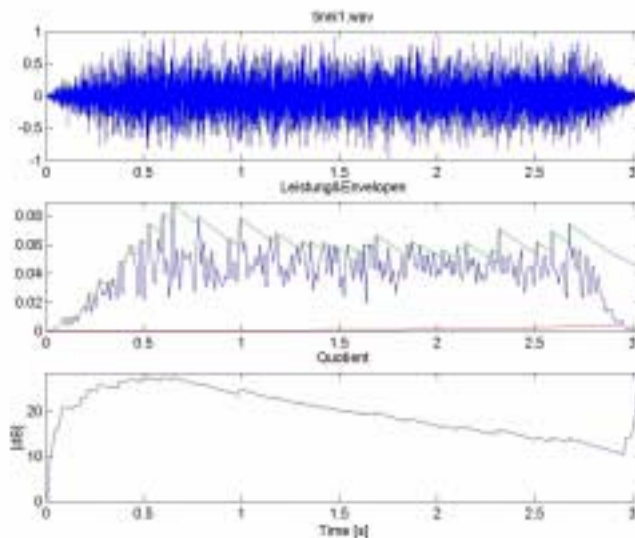
## 5.3 Geräusch



**Abb. 5.4:** Leistung Geräusch



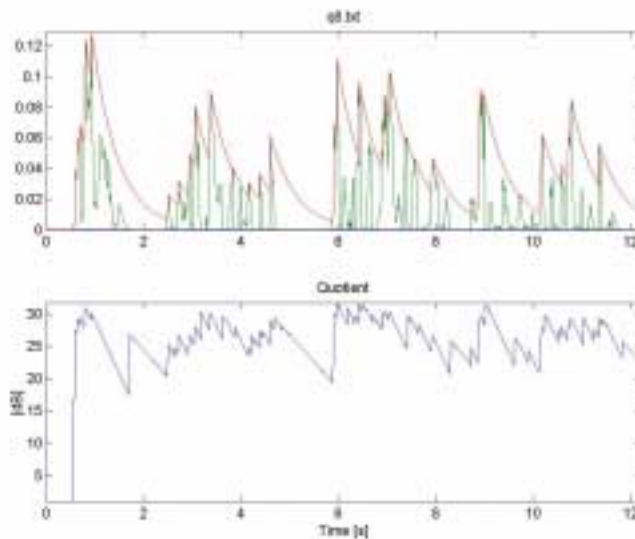
**Abb. 5.5:** *Envelopen Geräusch*



**Abb. 5.6:** *Quotient Geräusch*

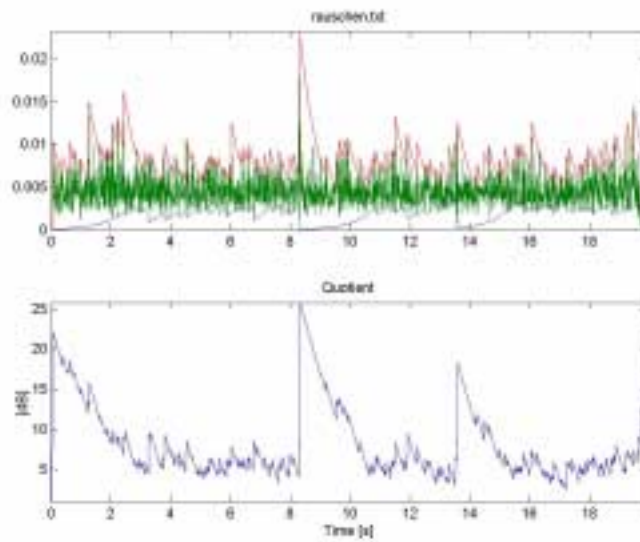
Bis jetzt wurden nur Signale berechnet, welche kürzer als drei Sekunden sind. Weil man, wie oben erläutert, die ersten paar Sekunden nicht bewerten darf, muss man längere Signale berechnen können. MATLAB ist dafür jedoch viel zu langsam und man würde sehr lange warten, bis ein brauchbares Resultat ausgegeben werden kann. Aus diesem Grund wurden die bisherigen Programme zusätzlich in der Hochsprache C++ implementiert (Siehe "Listing C++" auf Seite 64).

## 5.4 Längere Signale



**Abb. 5.7:** *Envelopen&Quotient Sprachsignal*

Bei einem längeren Sprachsignal sieht man gut, dass die untere Envelope gar nie wirklich steigen kann (Abb. 5.7). Es hat seinen Ursprung in der Dynamik der Sprache, welche die Leistung immer wieder mal klein werden lässt. Somit bleibt der Quotient ständig über 15 dB. Ganz anders sieht es bei einem Rauschsignal aus.  $E_{\text{Bottom}}$  steigt an und bildet die untere Envelope der Leistung. Der Quotient bleibt deshalb praktisch immer unter 30 dB (Abb. 5.8 Seite 23). Das Maximum nach ca. 8,2 Sekunden ist nur darauf zurückzuführen, dass die Leistung für kurze Zeit praktisch Null wird. Es handelt sich womöglich um einen Fehler des Ausgangssignals. Solche Fehler werden jedoch bei der Auswertung, ob ein Sprachsignal vorliegt oder nicht, mit der Autokorrelationsberechnung korrigiert (Siehe "Detektion" auf Seite 25).



**Abb. 5.8:** *Envelopen&Quotient Geräusch*

## 5.5 Rauschen überlagern

Mit der Überlagerung eines Rauschsignals mit einem Sprachsignal kann man die Effizienz dieser Methode weiter untersuchen.

Man sieht gut, dass in Abb. 5.10 Seite 24 die Dynamik der Sprache kleiner wird und somit auch der Quotient nicht mehr viel über 15 dB ist. Im Kapitel "Beispiele" auf Seite 29 wird weiter auf dieses Thema eingegangen.

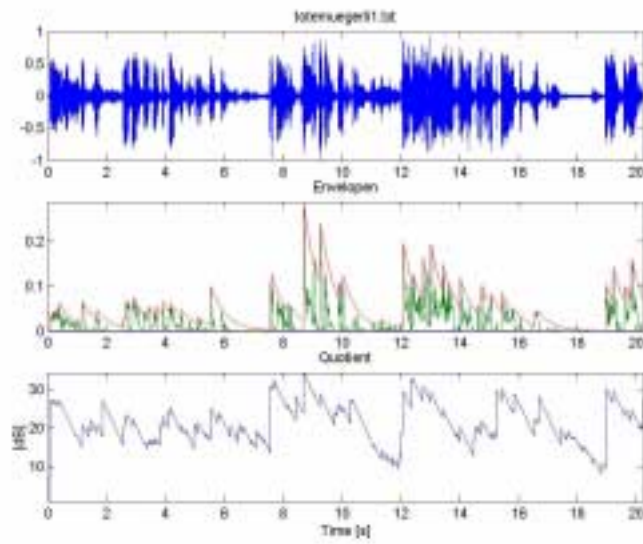


Abb. 5.9: Sprachsignal ohne Rauschen

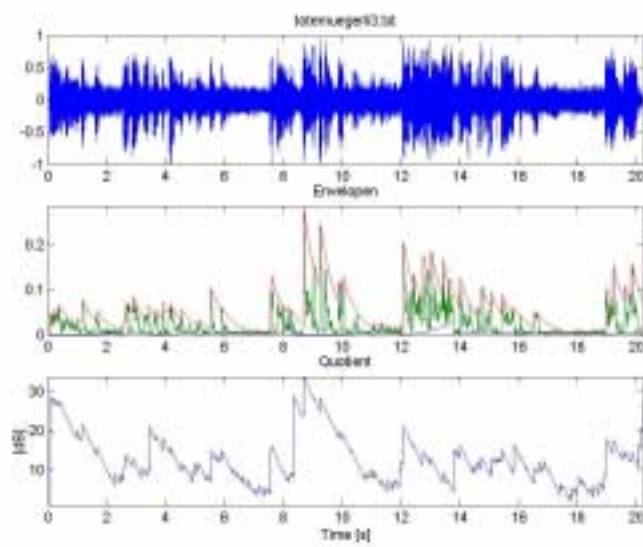


Abb. 5.10: Sprachsignal mit Rauschen

## 6 Detektion

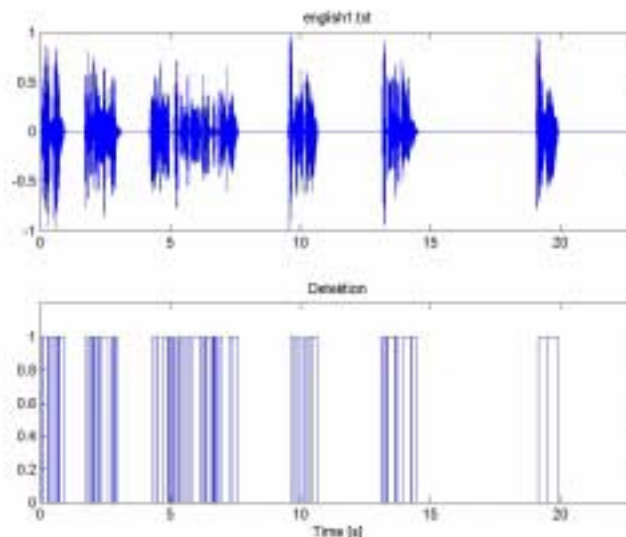
Nachdem über die zeitliche Verteilung der Signalleistung und die Autokorrelation entschieden werden kann, ob das Signal stimmhafte Anteile aufweist, gilt es, diese zwei Auswertungen sinnvoll zusammenzufügen, damit eine erhöhte Sicherheit in der Detektion gewährleistet werden kann. Dazu gibt es verschiedene Varianten:

### 6.1 Variante 1

Es wird je ein Schwellwert festgelegt.

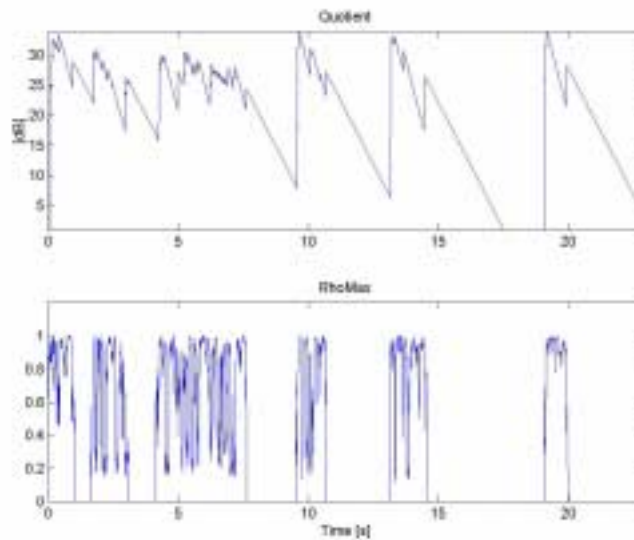
$$RhoMax > 0.8 \quad \&\& \quad Quotient > 15dB \quad \mathbf{6.1}$$

Da die maximale Autokorrelation bei den einzelnen Silben und Zischlauten stark ändert, wechselt die Detektion zwischen Sprache bzw. keine Sprache (beide Bedingungen erfüllt entspricht Sprache) innerhalb eines Wortes bzw. Satzes mehrmals (Abb. 6.1).



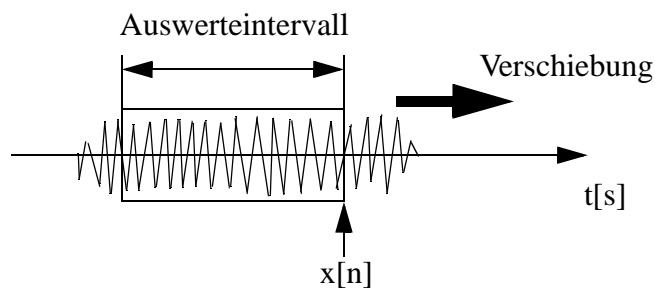
**Abb. 6.1:** Sprachsignal & Detektion

In der Abb. 6.2 ist das sprunghafte Verhalten der maximalen Autokorrelation ersichtlic.



**Abb. 6.2:** *Quotient & maximale Autokorrelation*

Es ist jedoch wünschenswert, dass aufeinanderfolgende Worte zusammen als Sprache detektiert werden. Somit muss ein längerer Zeitabschnitt zur Detektion betrachtet werden. Falls der stimmhafte Anteil überwiegt, wird das Signal als Sprache angenommen.



**Abb. 6.3:** *Detektion mit Fenster*

Der Nachteil dieser Variante ist die verspätete Reaktion bei einer Änderung im Signal. Die Verzögerung ist die Hälfte des betrachteten Zeitabschnittes. Eine Wahrscheinlichkeitsangabe macht in diesem Fall wenig Sinn, da der stimmhafte Anteil möglicherweise weit zurückliegt. Die Ausgabe würde sich also auf „1“ für Sprache und „0“ für stimmlose Abschnitte beschränken.

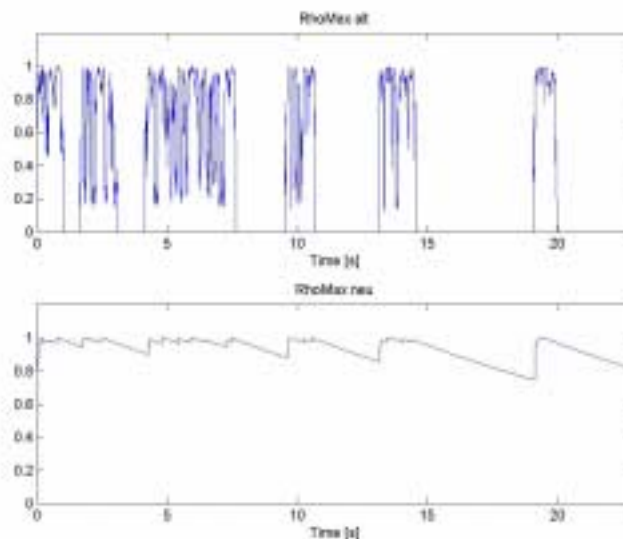
## 6.2 Variante 2

Die maximale Autokorrelation  $x[n]$  wird geglättet. Durch Bildung einer Envelope kann erreicht werden, dass die Sprünge nach Silben oder Zischlauten verschwinden.

$$e_{top}[n] = \max(x[n], e_{top}[n-1] \cdot top) \quad 6.2$$

$$top = \left( \frac{u_{Grenze}}{o_{Grenze}} \right)^{\frac{1}{5 \cdot T \cdot f_{abstast}}} \quad 6.3$$

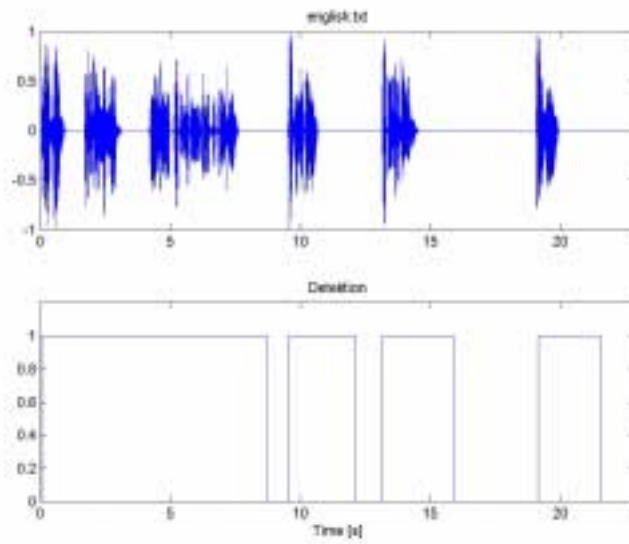
Der Faktor  $top$  errechnet sich aus der unteren- und oberen Grenze.  $o_{Grenze}$  ist eins, da die maximale Autokorrelation eins nie überschreitet. Die  $u_{Grenze}$  kann frei gewählt werden. Sie sollte nicht null sein, da der natürliche Logarithmus von null nicht definiert ist.  $u_{Grenze}$  wird bei  $1/1024$  angenommen. Die Envelope soll innerhalb 5 Zeitkonstanten  $T$  von der oberen- zur unteren Grenze abfallen. Für  $T$  wird eine Sekunde gewählt.



**Abb. 6.4:** Envelope der maximalen Autokorrelation

Für die Detektion gelten die gleichen Schwellwerte wie bei der ersten Variante. Durch die Envelope kann zu jedem Zeitpunkt abgefragt werden, ob die Bedingung erfüllt ist. Zudem können kurze Sprechpausen durch ein langsames Abfallen der Envelope überbrückt werden. Der grosse Vorteil liegt darin, dass sofort Sprache detektiert wird. Dafür ist wie bei der ersten Variante erst nach einer gewissen Zeit zu erkennen, dass es sich nicht mehr um Sprache handelt. Aufgrund dieser guten Eigenschaften wurde diese Methode in C++ implementiert (Siehe "Listing C++" auf Seite 64).

Nach Anwendung dieser Massnahme resultiert die Detektion wie in der Abb. 6.5 Seite 28. Die Verzögerung ist durch das Abfallen des Quotienten gegeben. Bei der Analyse der zeitlichen Verteilung der Signalleistung (Siehe "Signalleistung" auf Seite 17) werden ebenfalls Envelopen eingesetzt, welche sich nun auf die Verzögerung in der Detektion auswirken. Für stimmhafte Abschnitte ist die Ausgabe „1“ sonst „0“.



**Abb. 6.5:** *Detektion mit Envelope der maximalen Autokorrelation*

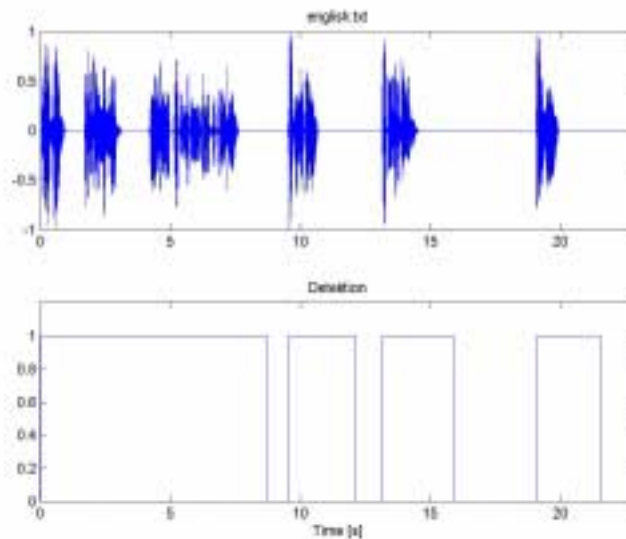
## 7 Beispiele

Die Beispiele sind in drei Gruppen unterteilt: Sprachsignale, Geräusche und gemischte Signale. Es wird jeweils das Signal, die Detektion, der Quotient und die maximale Autokorrelation eines Beispiels in Abhängigkeit der Zeit in Diagramme gezeichnet. Anhand dieser Angaben kann auf die Effizienz der Detektion geschlossen werden. Zu erwähnen ist, dass die Detektion in den ersten zwei Sekunden nur von der maximalen Autokorrelation abhängt (Siehe "Signalleistung" auf Seite 17). Die einzelnen Beispiele werden beurteilt, bei Fehlverhalten werden Verbesserungsvorschläge an die Detektion angebracht.

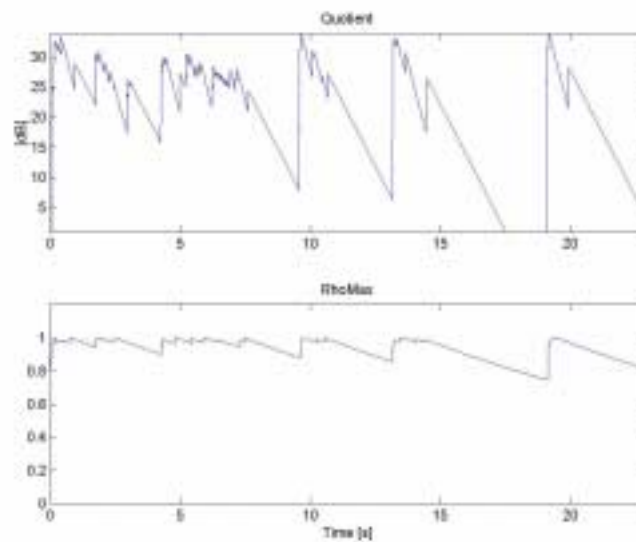
### Audio-Files:

Sprache	Geräusche	gemischte Signale
• english.wav	• jet.wav	• rummelplatz.wav
• ichbin.wav	• eierschlagen.wav	• lachen.wav
• jawohl.wav	• floete.wav	• babycriing.wav
• man.wav	• kriegslaerm.wav	• totemuegerli1.wav
• woman.wav	• pic.wav	• totemuegerli2.wav
• message.wav	• rauschen.wav	• totemuegerli3.wav
• tenoroper.wav	• sinus500hz.wav	• totemuegerli4.wav
• totemuegerli1.wav	• tinni1.wav	• totemuegerli5.wav
• zerror.wav	• tinni2.wav	

## 7.1 Sprachsignale

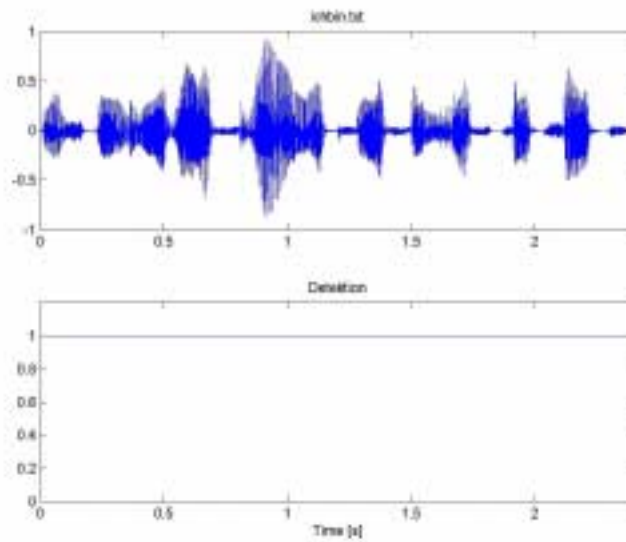


**Abb. 7.1:** Englische Sprache

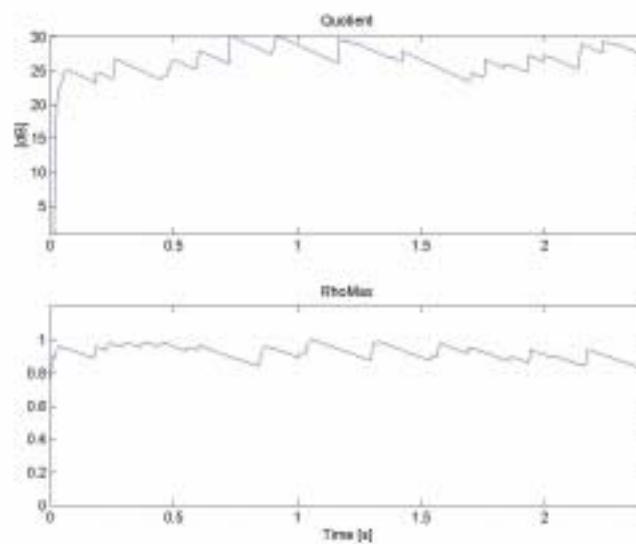


**Abb. 7.2:** Quotient & maximale Autokorrelation

Da sich die maximale Autokorrelation immer in der Nähe von eins bewegt und der Quotient gross ist, handelt es sich um Sprache. Wenn der Quotient unter 15 dB fällt werden auch die Sprechpausen angegeben. Das Abfallen des Quotienten könnte verlängert werden um längere Sprechpausen zu überbrücken.

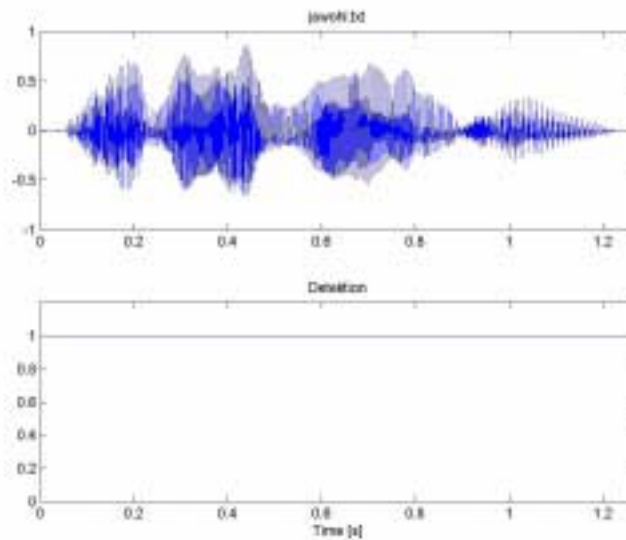


**Abb. 7.3:** „Ich bin Rudolf Ranekirch vom FCZ“

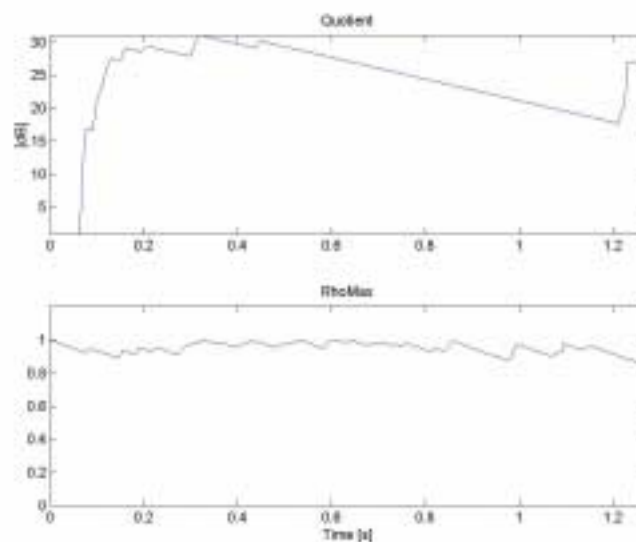


**Abb. 7.4:** Quotient & maximale Autokorrelation

Das kurze Sprachsignal ist praktisch nur von der maximalen Autokorrelation abhängig und wird somit als Sprache gedeutet.

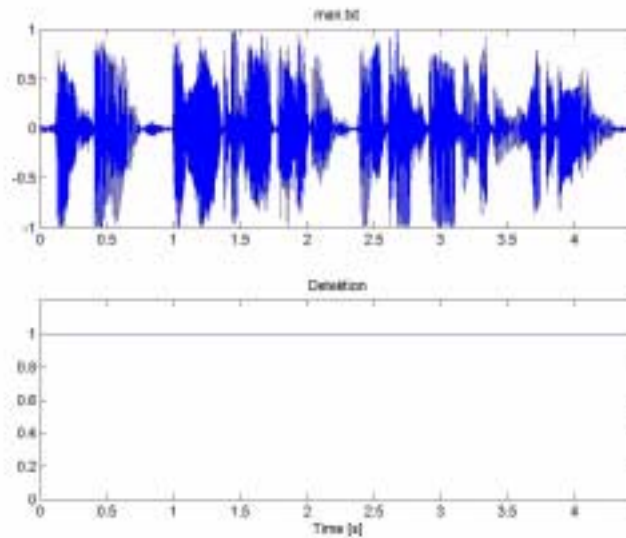


**Abb. 7.5:** „Jawohl hören sie“

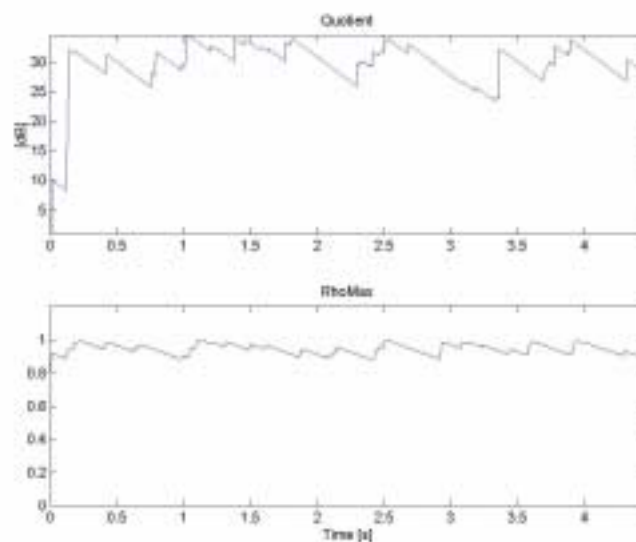


**Abb. 7.6:** Quotient & maximale Autokorrelation

Es handelt sich um ein kurzes Sprachsignal und deshalb nur von der maximalen Autokorrelation abhängig. In diesem Fall, wenn der Quotient überhaupt keinen Einfluss hat, könnte es sich auch um ein periodisches Signal handeln.



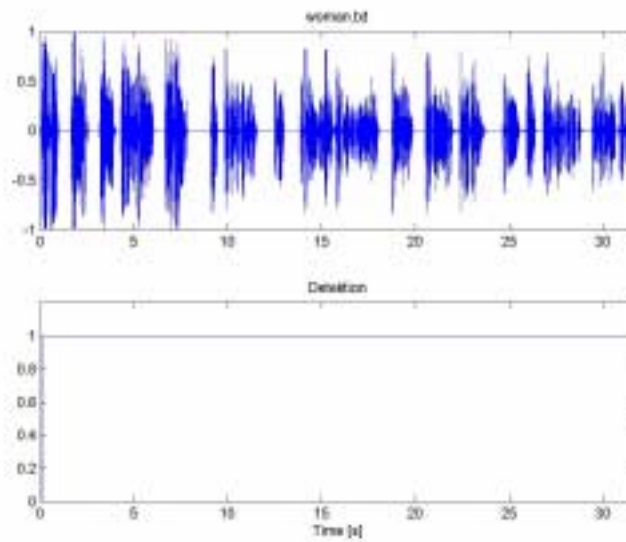
**Abb. 7.7:** Sprache eines Mannes



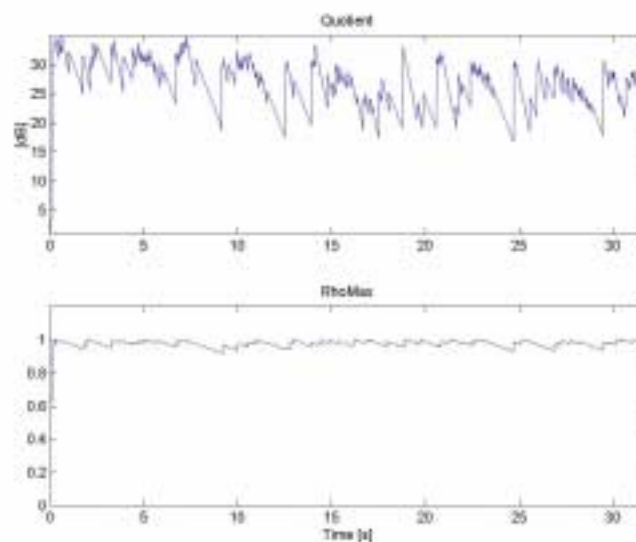
**Abb. 7.8:** Quotient & maximale Autokorrelation

Die Sprechpausen sind kurz und die maximale Autokorrelation nahe bei eins, deshalb wird das Signal immer als Sprachsignal detektiert.

Im Vergleich zur Frau besitzt der Mann einen grösseren Quotienten und eine trägere maximale Autokorrelation, da der Mann eine tiefere Pitchfrequenz und eine höhere Dynamik in der Signalleistung hat (vergl. Abb. 7.10 Seite 34).

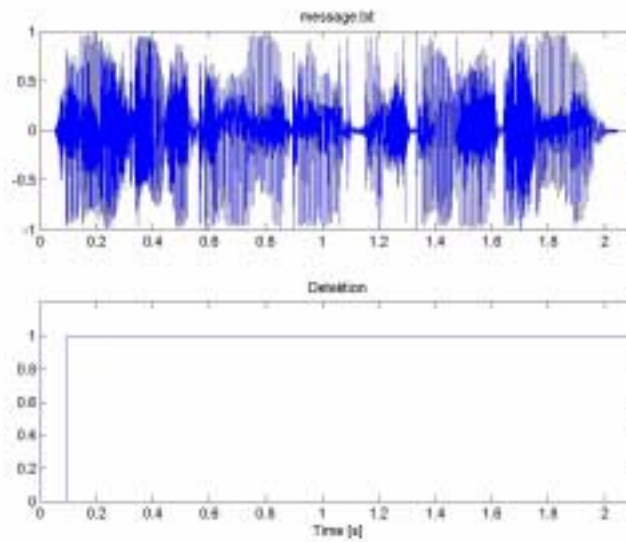


**Abb. 7.9:** Sprache einer Frau

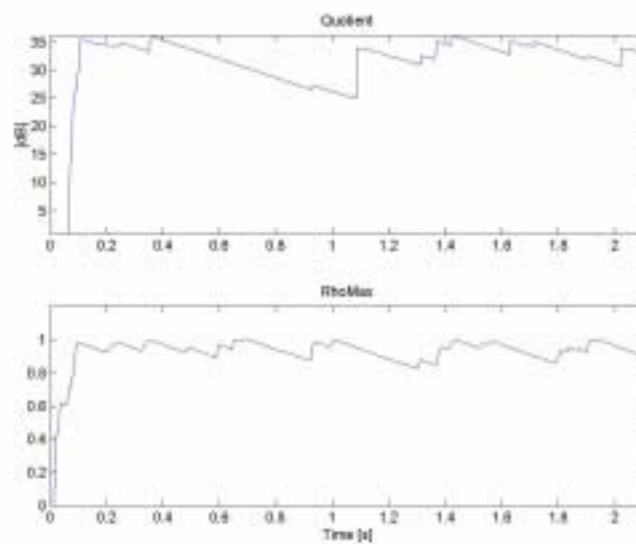


**Abb. 7.10:** Quotient & maximale Autokorrelation

Die Sprechpausen sind wiederum kurz, sodass über die ganze Dauer des Signals Sprache detektiert wird. Der Quotient ist ein bisschen kleiner als beim Mann, was auf die niedrigere Dynamik in Sprache zurückzuführen ist.

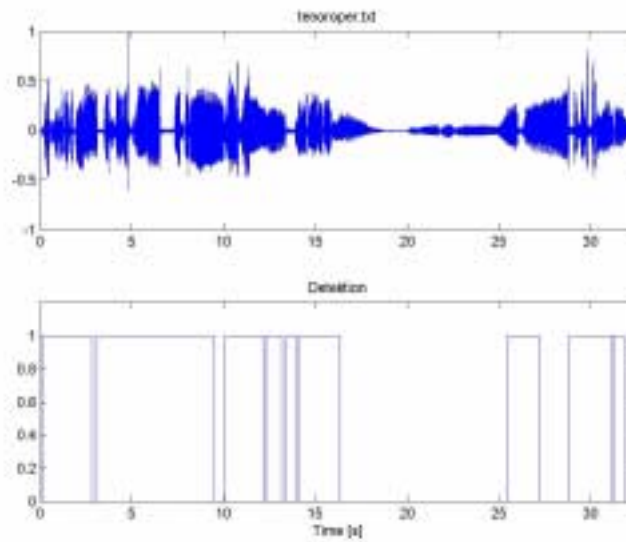


**Abb. 7.11:** „Lesen sie die Meldung auf dem Bildschirm“

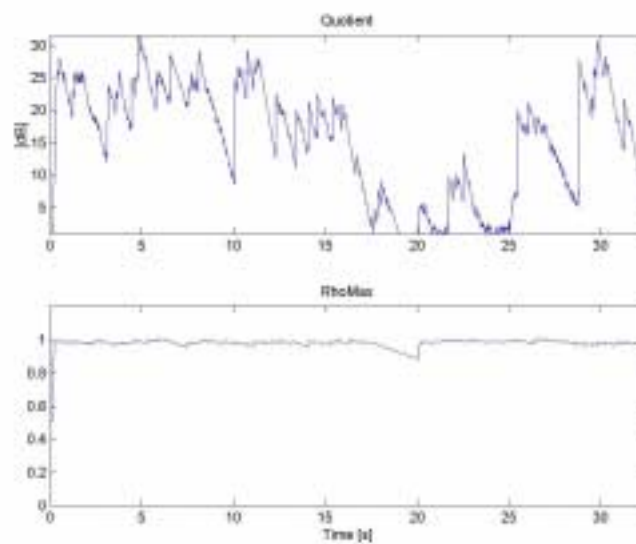


**Abb. 7.12:** Quotient & maximale Autokorrelation

Die maximale Autokorrelation ist der Grund, dass bei so kurzen Signalen über die ganze Zeit ein Sprachsignal detektiert wird.

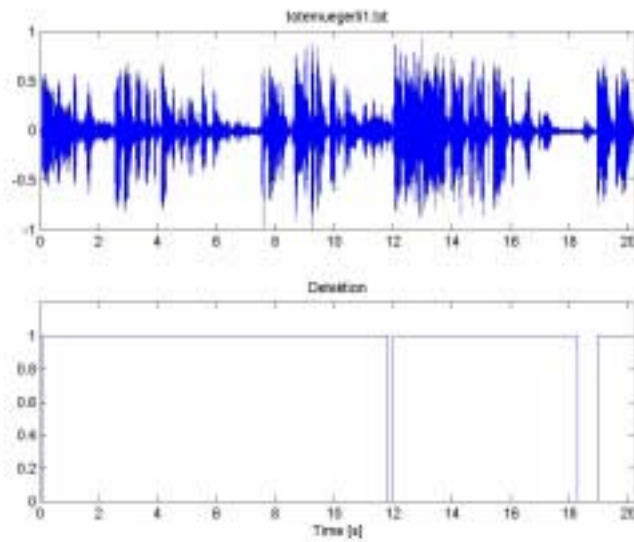


**Abb. 7.13:** *Operngesang*

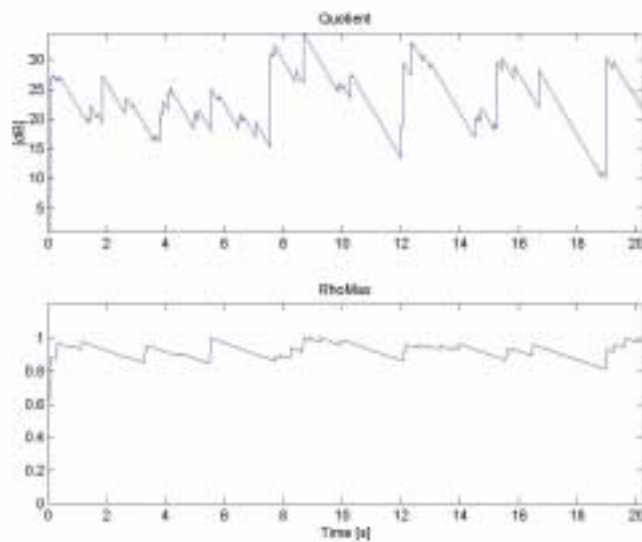


**Abb. 7.14:** *Quotient & maximale Autokorrelation*

Bei einer gleichmässigen Signalleistung, fällt der Quotient unter 15 dB, obwohl der Gesang von einer Menschenstimme erzeugt wurde. Bei Sprache bleibt die Leistung normalerweise nicht über längere Zeit konstant.



**Abb. 7.15:** *S-Totemügerli*



**Abb. 7.16:** *Quotient & maximale Autokorrelation*

S-Totemügerli ist eine Erzählung mit zwei längeren Sprechpausen. Wegen dem Quotient, der bei Sprechpausen kleiner wird, geht die Detektion zwischendurch auf Null. Mit einer grösser gewählten Zeitkonstanten (Siehe "Signalleistung" auf Seite 17) erreicht man, dass auch längere Sprechpausen als Sprache erkannt werden.

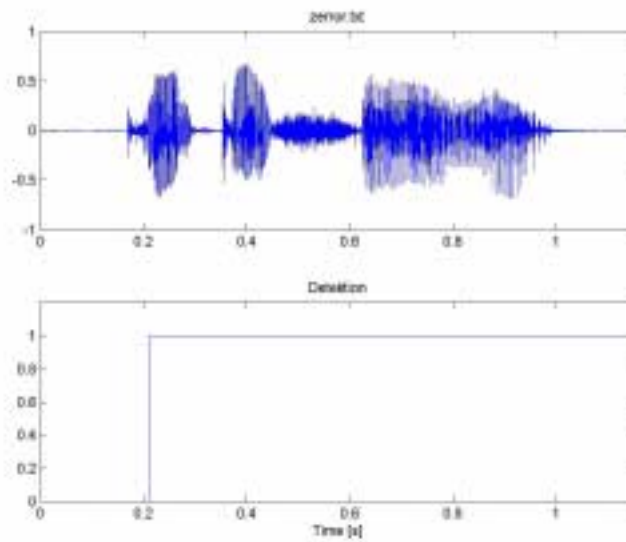


Abb. 7.17: „Druckerfehler“

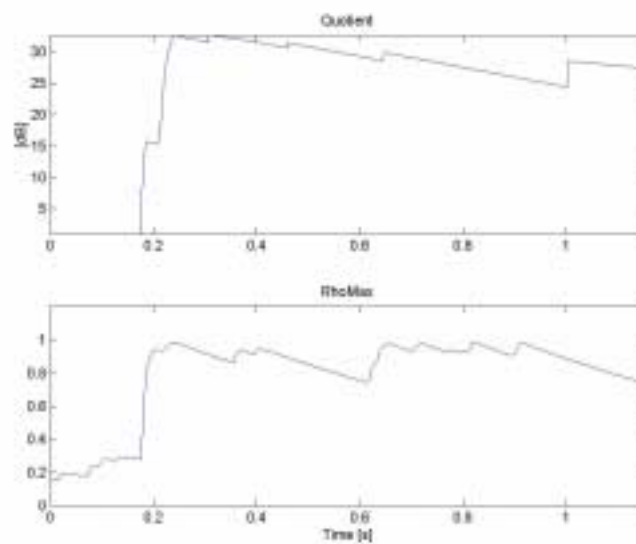


Abb. 7.18: Quotient & maximale Autokorrelation

Das Ergebnis dieses kurzen Signals ist nur von der maximalen Autokorrelation abhängig.

## 7.2 Geräusche

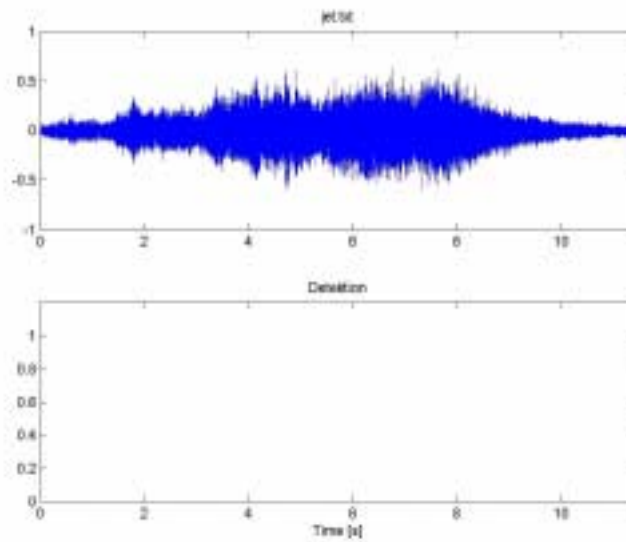


Abb. 7.19: *Jet*

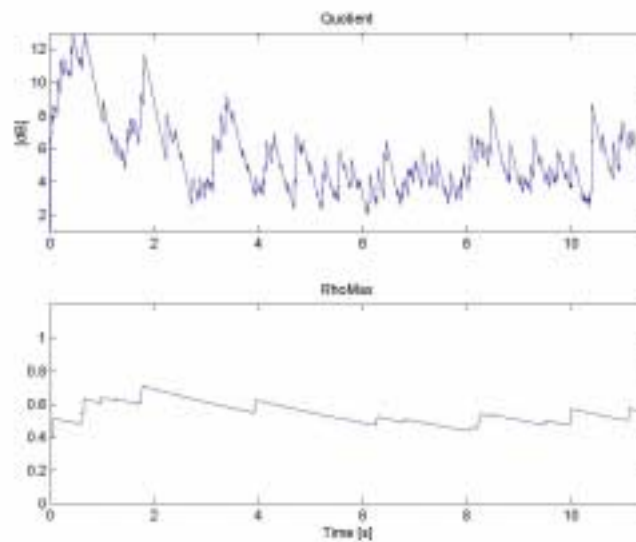
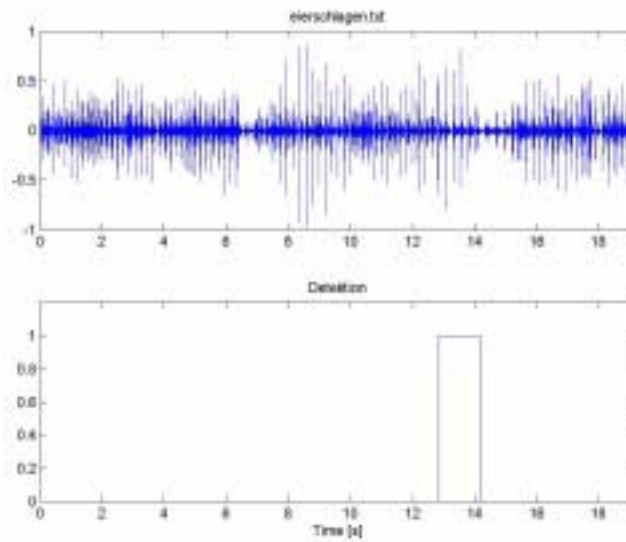
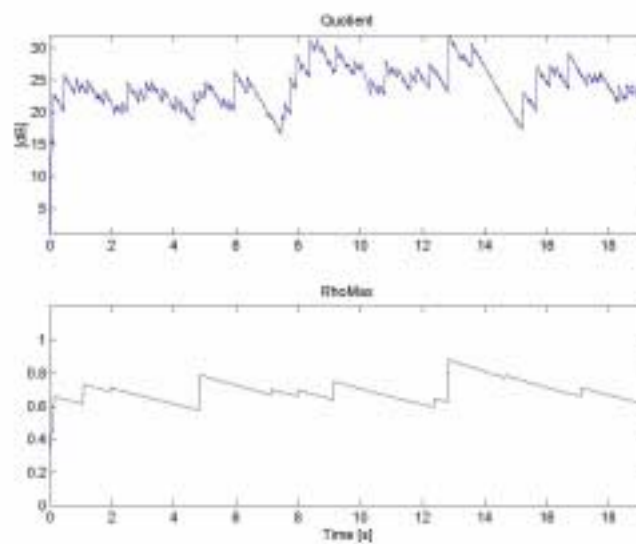


Abb. 7.20: *Quotient & maximale Autokorrelation*

Hier handelt es sich um einen startenden Jet. Das Geräusch ist weder periodisch noch ist eine hohe Dynamik zu erkennen. Es handelt sich deshalb um ein Geräusch.

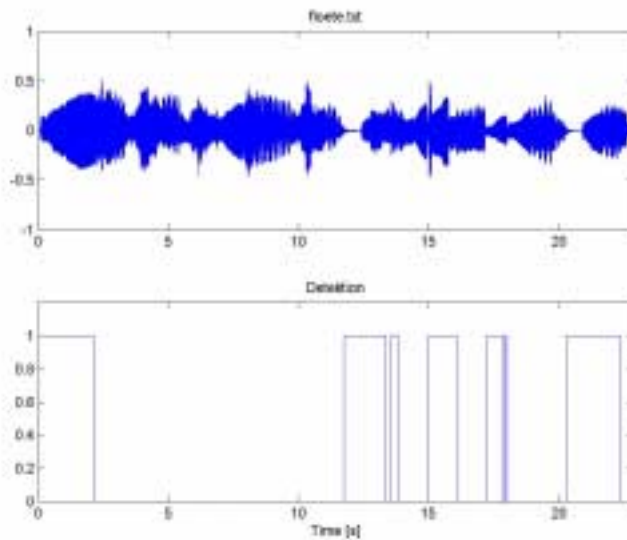


**Abb. 7.21:** Eierschlagen

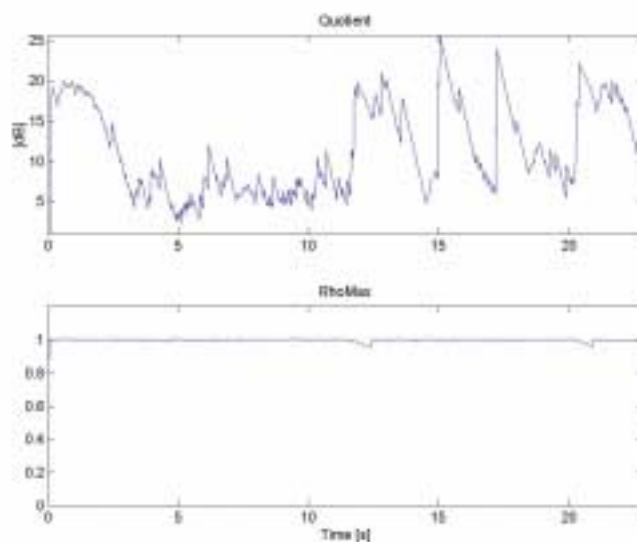


**Abb. 7.22:** Quotient & maximale Autokorrelation

Das Signal weist eine hohe Dynamik in der Signalleistung auf. Die maximale Autokorrelation ist nicht so hoch wie bei einem Sprachsignal. Eierschlagen kann also als Geräusch gedeutet werden.

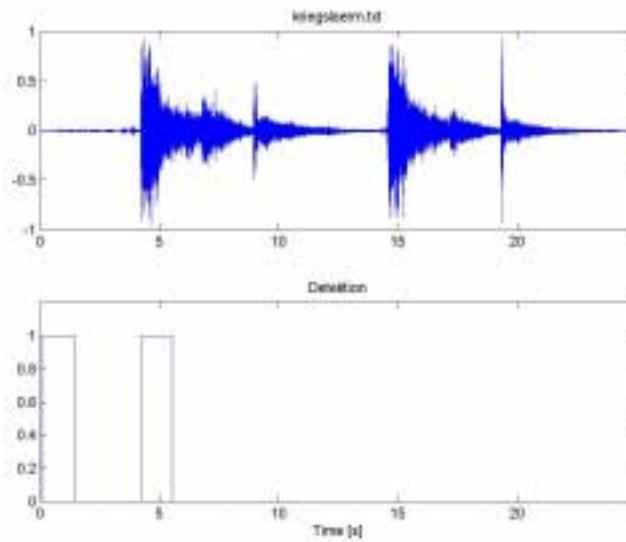


**Abb. 7.23:** *Flöte*

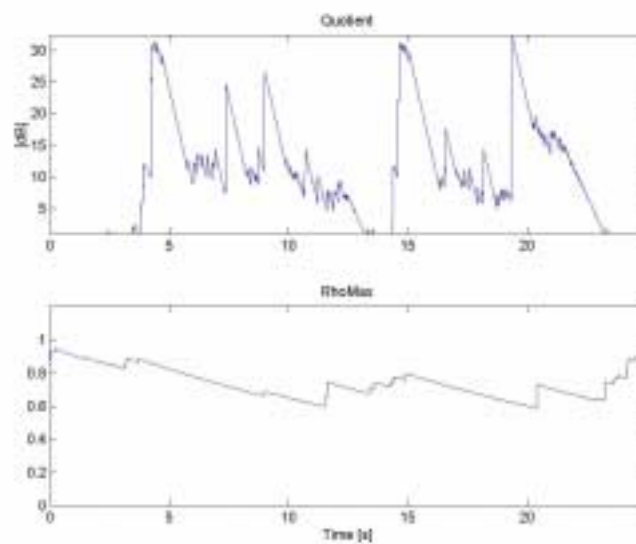


**Abb. 7.24:** *Quotient & maximale Autokorrelation*

Da das Signal periodisch ist, hat es eine hohe maximale Autokorrelation. Der Quotient weist zum Teil eine ähnliche Dynamik wie Sprache auf. Das Signal wird als Sprache gedeutet, obwohl es sich um ein Musiksignal handelt. Man müsste ein weiteres Entscheidungskriterium in die Detektion einfließen lassen, um auch solche Fälle abzudecken (Siehe "Erweiterungsmöglichkeit" auf Seite 56).

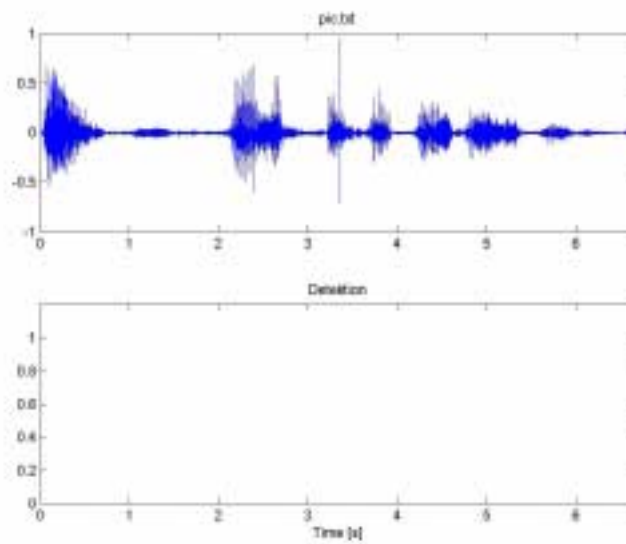


**Abb. 7.25:** *Kriegslärm*

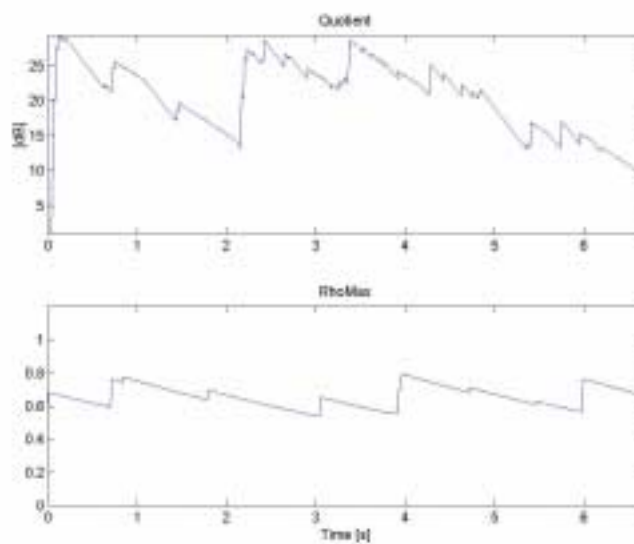


**Abb. 7.26:** *Quotient & maximale Autokorrelation*

Zu Beginn ist leise Sprache zu hören, daher resultiert eine hohe maximale Autokorrelation. Bei einer Detonation macht der Quotient einen grossen Sprung. Deshalb die Sprachdetektion nach ca. fünf Sekunden.



**Abb. 7.27:** Schweinegrunzen



**Abb. 7.28:** Quotient & maximale Autokorrelation

Obwohl der Quotient auf Sprache deuten würde, ist das Signal aufgrund der niedrigen maximalen Autokorrelation kein Sprachsignal. Die Pitchfrequenz liegt anscheinend unter der tiefsten in der Berechnung erwarteten Frequenz (Siehe "Autokorrelation" auf Seite 7).

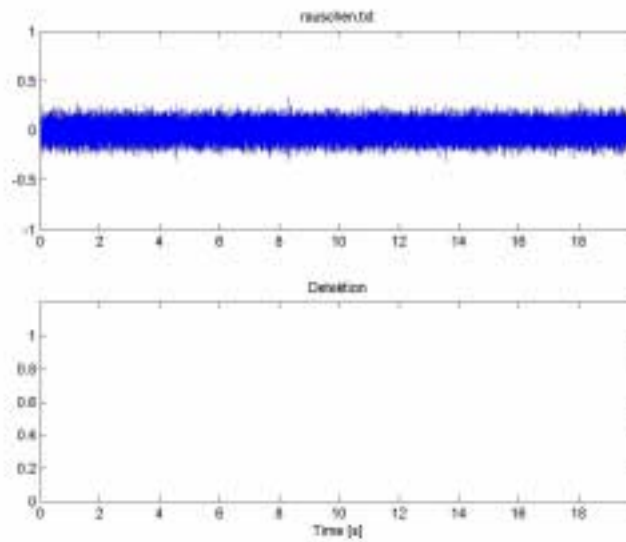


Abb. 7.29: Rauschen

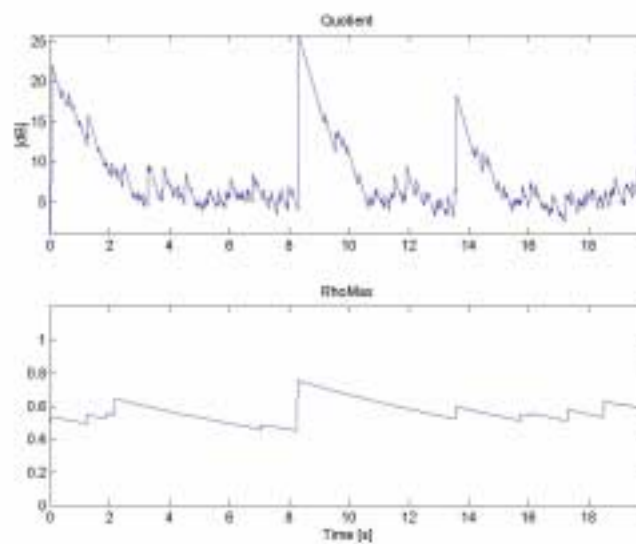
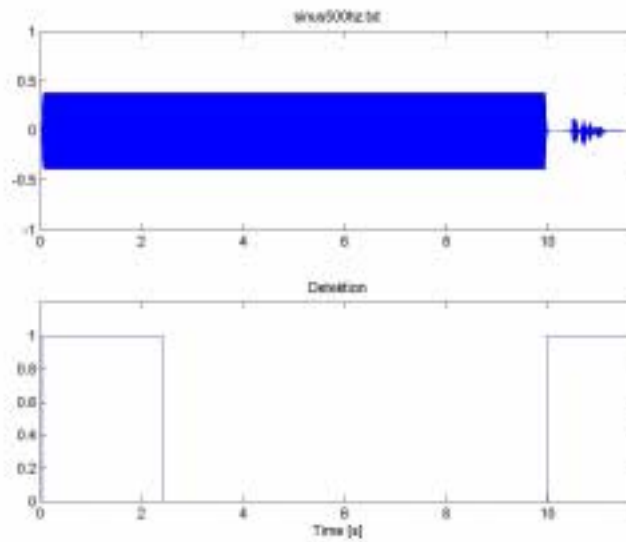
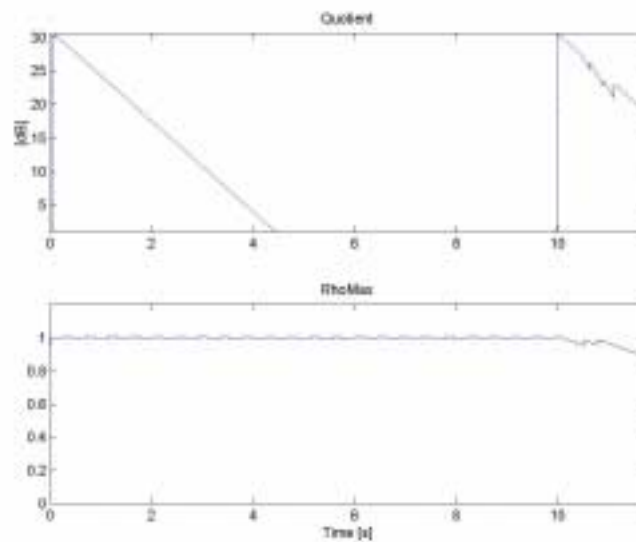


Abb. 7.30: Quotient & maximale Autokorrelation

Dieses Geräusch wurde schon im Kapitel 5.4 "Längere Signale" auf Seite 22 behandelt.



**Abb. 7.31:** *Sinus 500Hz*



**Abb. 7.32:** *Quotient & maximale Autokorrelation*

Gerade bei periodischen Signalen ist man auf den Quotienten angewiesen. Da dieser aber erst nach zwei Sekunden in die Detektion einfließt und zuerst abklingen muss, wird der Sinus am Anfang als Sprache detektiert.

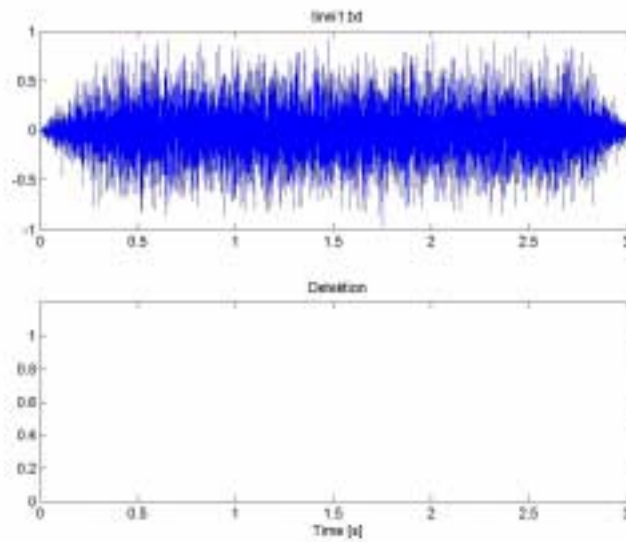


Abb. 7.33: *Tinnitus*

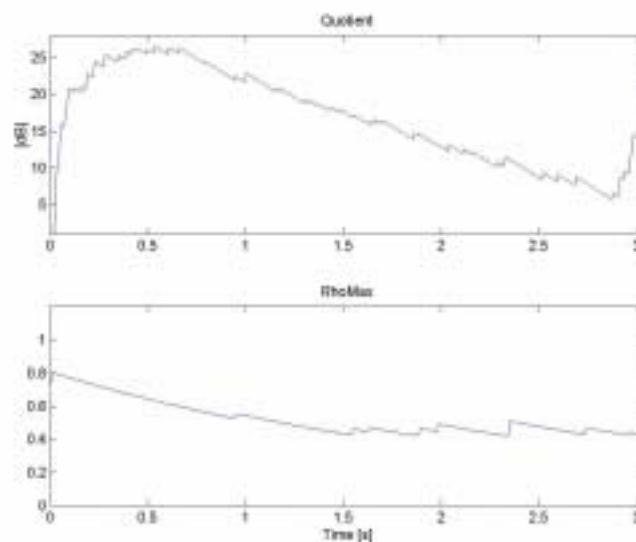


Abb. 7.34: *Quotient & maximale Autokorrelation*

Die maximale Autokorrelation liegt deutlich unter dem Schwellwert von 0,8. Einzig der Quotient ist zu Beginn deutlich über dem Schwellwert von 15dB (vergl. "Signalleistung" auf Seite 17). Dieser hat jedoch keinen Einfluss, er wird die ersten zwei Sekunden sowieso nicht berücksichtigt. Die erste Variante des Tinnitus ist kein Sprachsignal.

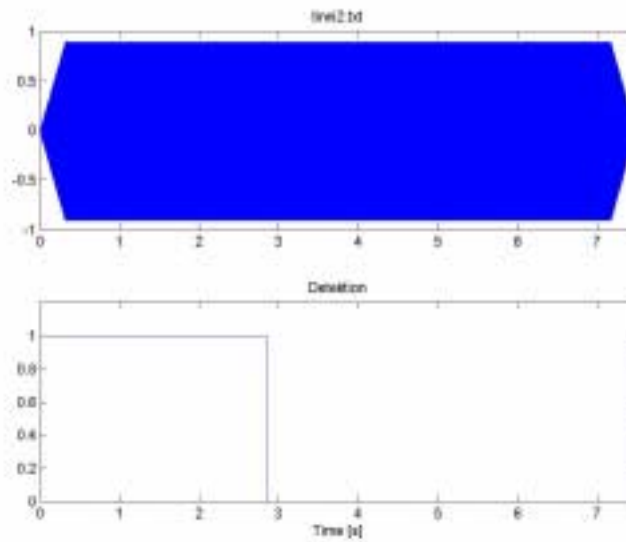


Abb. 7.35: *Tinnitus*

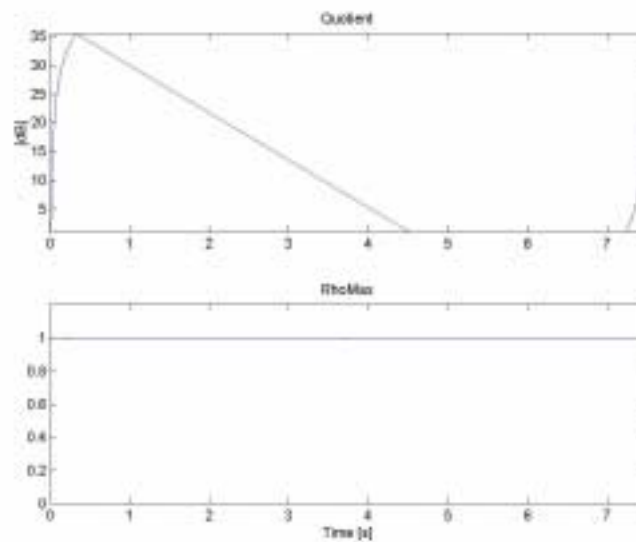


Abb. 7.36: *Quotient & maximale Autokorrelation*

Die zweite Variante des Tinnitus wird die ersten paar Sekunden als Sprachsignal gedeutet. Das Signal ist periodisch. Es ist aber keine hohe Dynamik vorhanden, trotzdem muss der Quotient zuerst abfallen. In diesem Fall müsste man länger warten und sich am Anfang nicht nur auf die Autokorrelation verlassen, bis das Signal detektiert werden kann.

### 7.3 Gemischte Signale

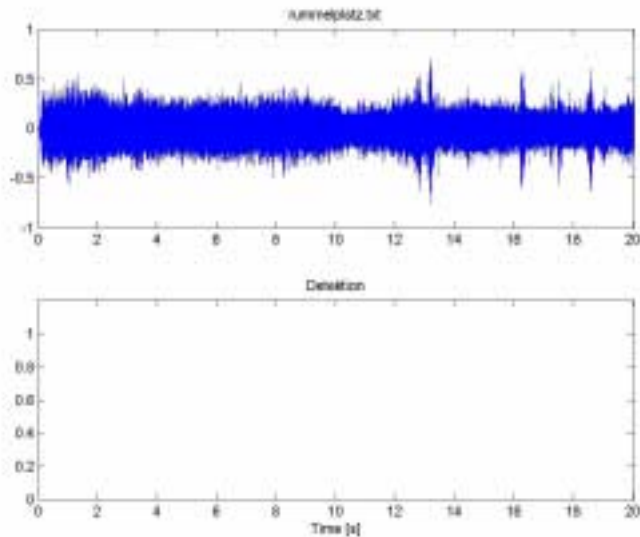


Abb. 7.37: Rummelplatz

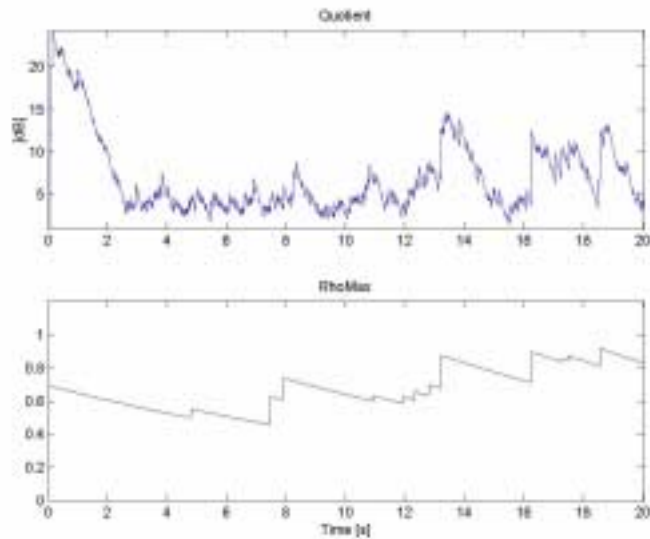
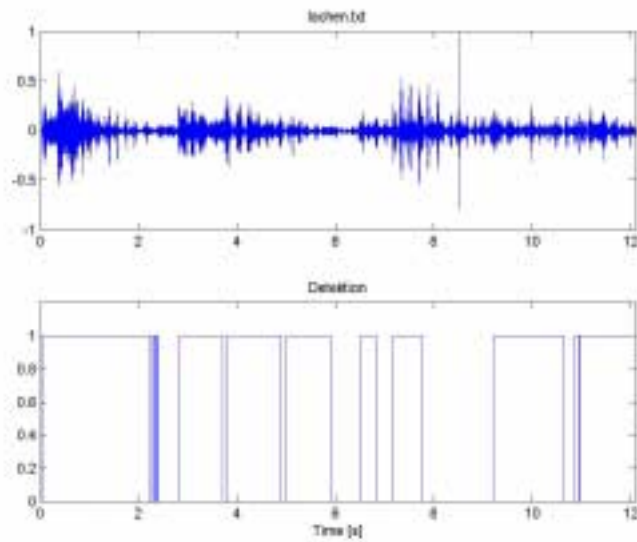
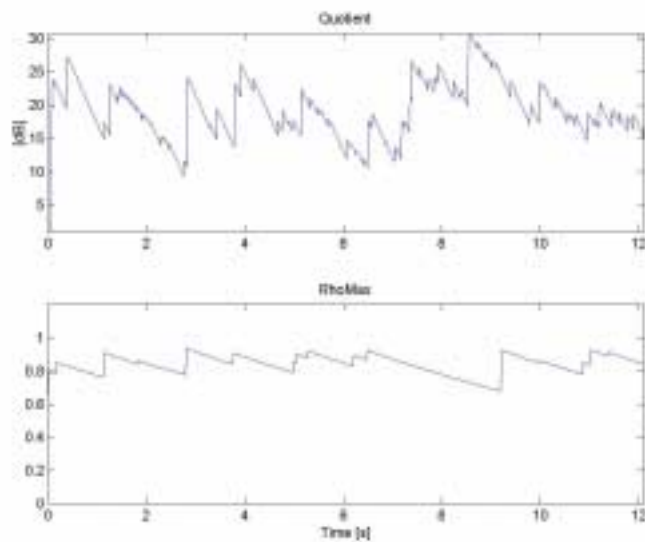


Abb. 7.38: Quotient & maximale Autokorrelation

Kreischende Kinder und ein lautes Rauschen bewirken eine niedrige Dynamik. Die maximale Autokorrelation überschreitet den Schwellwert von 0,8 erst gegen Ende des Signals. Wegen dem kleinen Quotienten hat sie jedoch keinen Einfluss auf die Detektion.

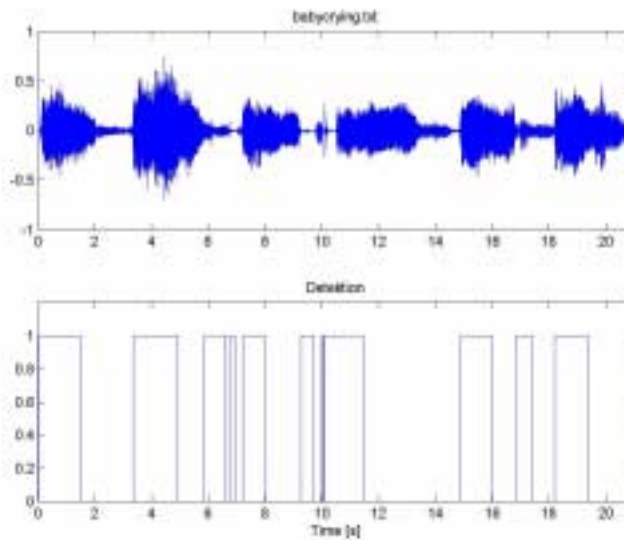


**Abb. 7.39:** *Lachen*

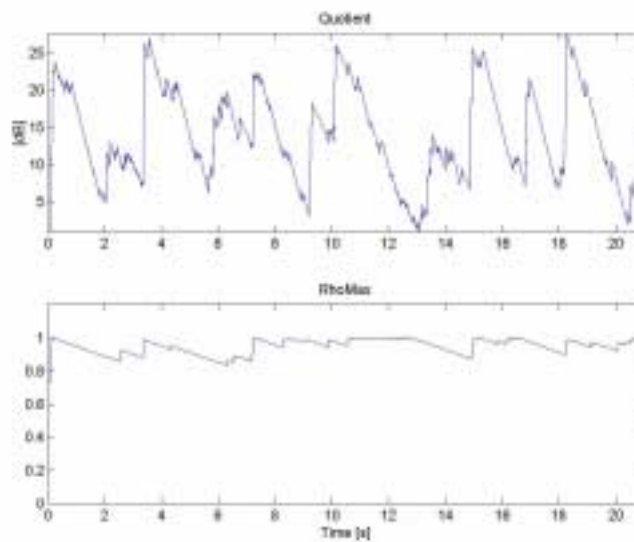


**Abb. 7.40:** *Quotient & maximale Autokorrelation*

Gehört Lachen zur Sprache? Es ist sicherlich ein Grenzfall, wie man der Detektion entnehmen kann. Die maximale Autokorrelation liegt immer in der Nähe von 0,8. Die Detektion ist also vom Quotienten und von der maximalen Autokorrelation abhängig (vergl. „weinendes Baby“ auf der nächsten Seite).

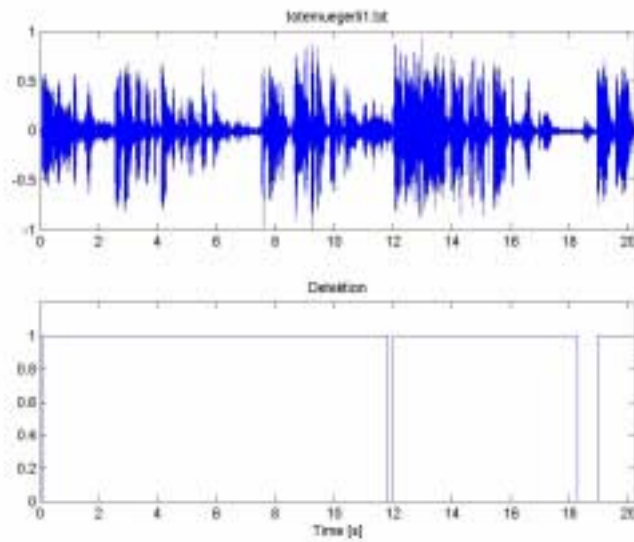


**Abb. 7.41:** *Weinendes Baby*

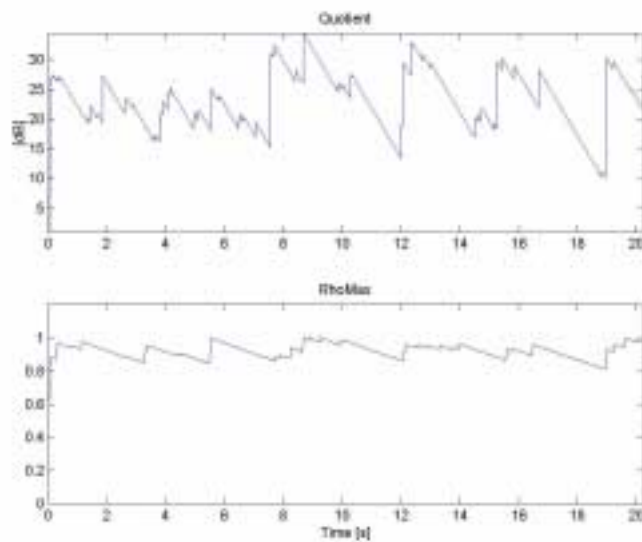


**Abb. 7.42:** *Quotient & maximale Autokorrelation*

Auch ein weinendes Baby liegt auf der Grenze zwischen Sprache und nicht Sprache. Die maximale Autokorrelation befindet sich während der ganzen Dauer des Signals über dem Schwellwert von 0,8. Die Detektion ist also nur vom Quotienten abhängig.

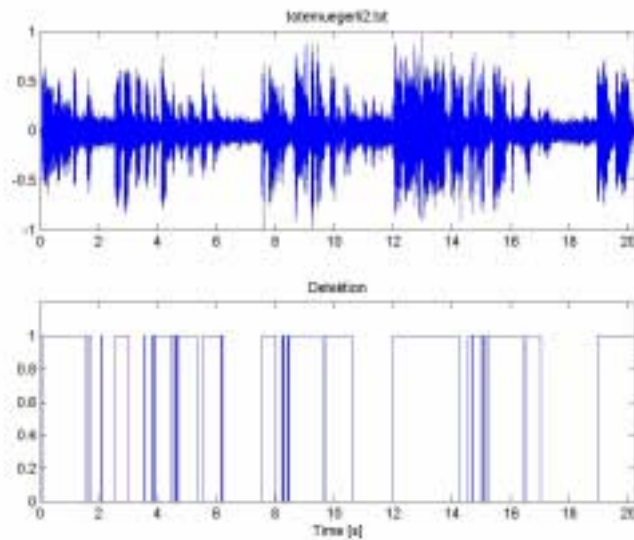


**Abb. 7.43:** *S-Totemügerli ohne Rauschen*

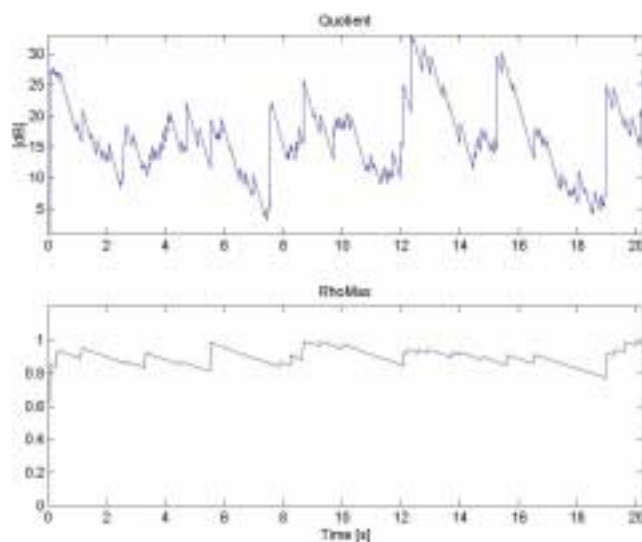


**Abb. 7.44:** *Quotient & maximale Autokorrelation*

Nachfolgend ist das „Totemügerli“ vier mal aufgeführt. Jedesmal mit einem stärkeren Rauschen überlagert. Je mehr Rauschen überlagert wird, desto kleiner wird der Signal-Rauschabstand (Berechnung Signal-Rauschabstand siehe Kapitel A.6 „signalrausch.m“ auf Seite 63). In Abb. 7.43 ist noch kein Rauschen überlagert. Es wird während der ganzen Signaldauer Sprache detektiert.

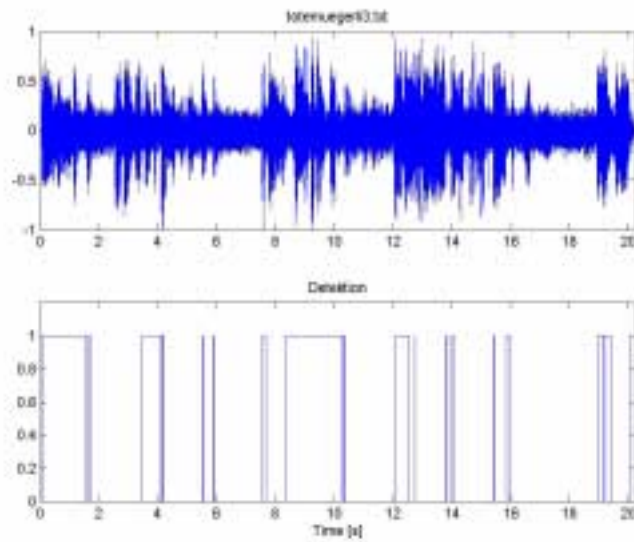


**Abb. 7.45:** *S*'Totemügerli mit Rauschen

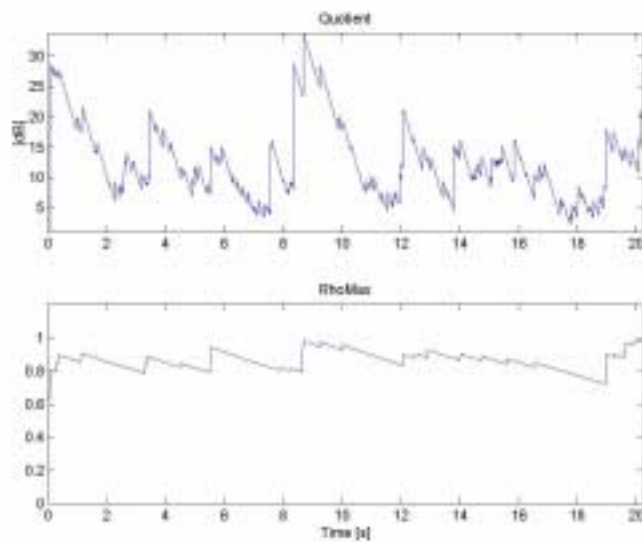


**Abb. 7.46:** *Quotient & maximale Autokorrelation*

Der Signal-Rauschabstand beträgt 11,4 dB. Die maximale Autokorrelation bewegt sich immer noch deutlich über dem Schwellwert von 0,8. Der Quotient hat sich verkleinert. Die Detektion macht viele Sprünge.

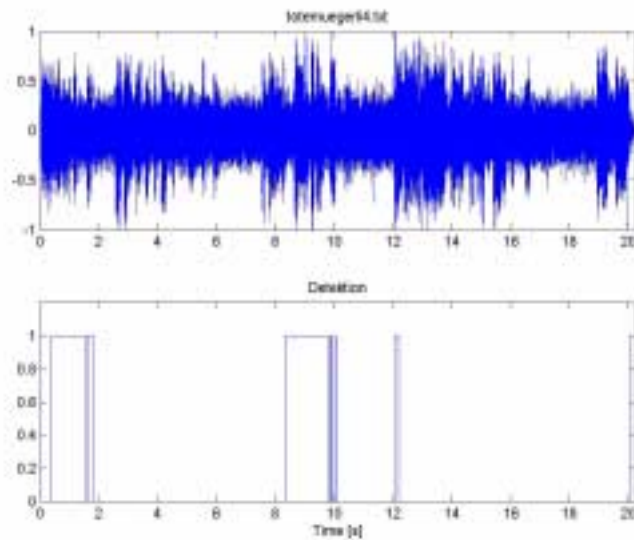


**Abb. 7.47:** *S*'Totemügerli mit Rauschen

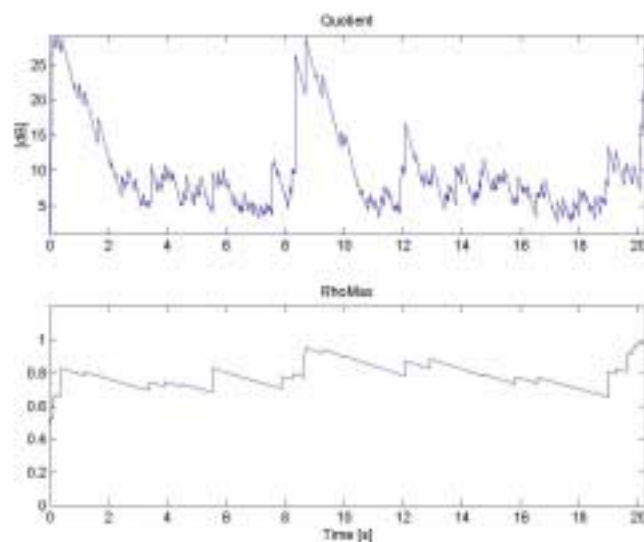


**Abb. 7.48:** *Quotient & maximale Autokorrelation*

Der Signal-Rauschabstand beträgt 6,6 dB. Die maximale Autokorrelation bewegt sich immer noch über dem Schwellwert. Der Quotient verkleinert sich weiter. Die Abstände zwischen Sprache und nicht Sprache in der Detektion vergrössern sich.

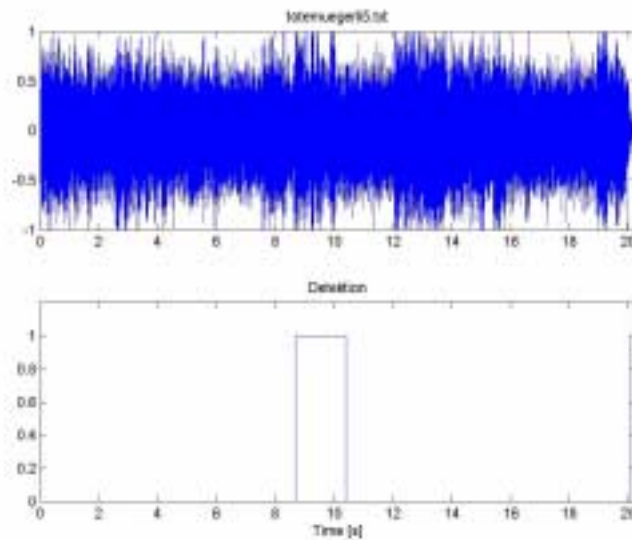


**Abb. 7.49:** *S* Totemügerli mit Rauschen

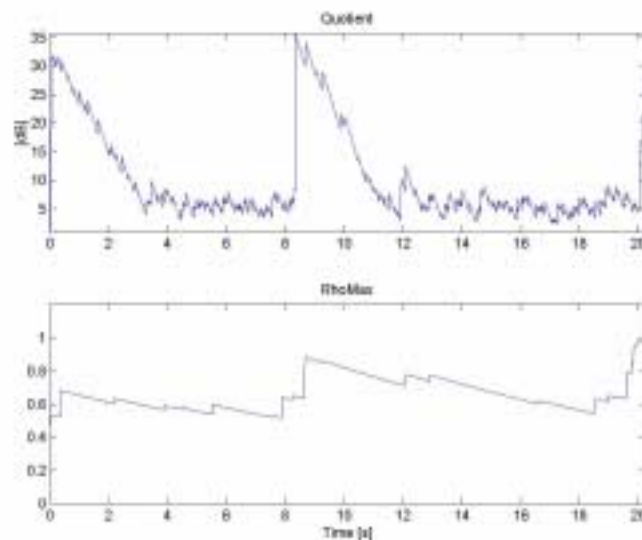


**Abb. 7.50:** *Quotient & maximale Autokorrelation*

Der Signal-Rauschabstand beträgt 1,5 dB. Die maximale Autokorrelation ist ein bisschen kleiner und bewegt sich zwischendurch auch unter dem Schwellwert. Der Quotient verkleinert sich weiter. Es wird nur noch für kurze Zeit Sprache detektiert.



**Abb. 7.51:** *S*'Totemügerli mit Rauschen



**Abb. 7.52:** *Quotient & maximale Autokorrelation*

Der Signal-Rauschabstand beträgt  $-3,4$  dB. Die Leistung des Rauschens ist grösser wie die Leistung des Signals. Zum ersten Mal sinkt auch die maximale Autokorrelation dauernd unter den Wert von  $0,8$ . Der Quotient ist bis auf zwei Impulse mit nachfolgendem einschwingen sehr klein. Es wird kaum mehr Sprache detektiert.

## 8 Erweiterungsmöglichkeit

- Verbessern der Detektion, damit auch kritische Signale richtig erkannt werden
- die einzelnen Programme so ändern, dass die Berechnungszeiten wesentlich kürzer werden
- weiteres Verfahren, z.B. Cepstral-Verfahren, untersuchen und implementieren
- DSP einsetzen
- GUI erstellen

## 9 Schlusswort

Nachdem wir uns in MATLAB eingearbeitet haben, beschäftigten wir uns mit den Merkmalen der Sprache. Danach versuchten wir diese Erkenntnisse in die Sprachdetektion einfließen zu lassen. Nach Absprache mit unserem Betreuer wählten wir, wie im Bericht ersichtlich, zwei Methoden. Um die Berechnungszeit zu verkürzen, haben wir die Detektion in C++ implementiert. Da es uns an Routine fehlte, investierten wir viel Zeit in diesem Bereich. Dabei konnten wir sehr viel profitieren. Weiter mussten wir die Detektion mehrmals überarbeiten, bis sie wunschgemäß funktionierte. Die Erfahrung mit anderen Schreibprogrammen, bewegte uns dazu, die Dokumentation mit dem Programm FrameMaker zu schreiben. Dieser Entscheid hat sich als gut erwiesen.

Zusammenfassend kann man sagen, dass die erste Studienarbeit für uns sehr lehrreich und interessant war. An dieser Stelle möchten wir auch unserem Betreuer A. Schaub recht herzlich danken.

Rapperswil, 1. März 2002

Florian Meier

Alexander Schläpfer

## 10 Literaturverzeichnis

- I Panos E. Papamichalis: Practical Approaches to Speech Coding, Prentice Hall, Englewood Cliffs, New Jersey, 1987.
- II L.R. Rabiner, B.H. Juang: Fundamentals of Speech Recognition, Prentice Hall, Englewood Cliffs, New Jersey, 1996.

## A Listing MATLAB

- p\_track.m
- leist.m
- center\_clipping.m
- center\_clipping1.m
- ausgabe.m
- signalrausch.m

### A.1 p\_track.m

```
%Funktion zur Erkennung der Pitchfrequenz und berechnung der Autokorrelation
%Der Funktion muss ein Filename übergeben werden
%Aufruf: p_track('file');
%*****
function rxx=p_track(file)

signal=wavread(file);
%signal=center_clipping1(signal);

f_abtast=22050;
f_grenz=800;

%Tiefpassfilter
[B,A]=butter(6,f_grenz/(f_abtast/2));
signal=filter(B,A,signal);

size=length(signal);
t=[0:1/f_abtast:size/f_abtast-1/f_abtast];
Tmin=30;
Tmax=300;
N=1200;
time=Tmax+N;
rxx=[];
rho=[];
delta_rxx=[];
rhomax=[];
maxpos=[];
y_wert=time;
x_wert=y_wert-Tmin;
schrittweite=100;
x=0;
y=0;

while y_wert <= size

    y=signal(y_wert-N+1:y_wert);
    delta_rxx=0;
    sq_y=sum(y.^2);

    while x_wert >= y_wert-Tmax
        x=signal(x_wert-N+1:x_wert);
        rho=sum(x.*y)/(sqrt(sum(x.^2)*sq_y));
        delta_rxx=[delta_rxx,rho];
        x_wert=x_wert-1;
    end

end
```

```

[a b]=max(delta_rxx);
rhomax=[rhomax,a];
maxpos=[maxpos,b];
rxx=[rxx,delta_rxx];
time=time+schrittweite;
y_wert=time;
x_wert=y_wert-Tmin;
end
rhomax=[zeros(1,fix((Tmax+N)/schrittweite)-1),rhomax];
size_rhomax=length(rhomax);
t_rhomax=[0:size/(f_abtast*size_rhomax):size/f_abtast-1/f_abtast];

pitch=f_abtast./(maxpos+Tmin);
pitch=[zeros(1,fix((Tmax+N)/schrittweite)-1),pitch];
size_pitch=length(pitch);
t_pitch=[0:size/(f_abtast*size_pitch):size/f_abtast-1/f_abtast];

figure(1)
subplot(311);
plot(t,signal)
title('Signal');
axis([0 size/f_abtast-1/f_abtast -1 1]);

subplot(312);
plot(t_rhomax,rhomax)
title('rho max');
axis([0 size/f_abtast-1/f_abtast 0 1]);

subplot(313);
plot(t_pitch,pitch)
title('Pitch');
axis([0 size/f_abtast-1/f_abtast 0 800]);
ylabel('Pitchfrequenz [Hz]');
xlabel('Zeit [s]');

```

## A.2 center\_clipping1.m (Variante $A_{max}$ )

```

%Funktion für die Berechnung der Enevelope und danach Streckung des Signals
%Der Funktion muss der Signalvektor übergeben werden
%Funktion ist nur innerhalb von p_track aufzurufen
%Aufruf: center_clipping1('signalvektor');
%*****
function signal_cc=center_clipping1(signal)
size=length(signal);
g=0.4;
window=100;
i=1;
while i<=size
    max=0;

    %max bestimmen
    for k=0:window
        if i+k<=size
            if max<signal(i+k)
                max=signal(i+k);
            end
        end
    end

    %g*max ausschneiden
    for l=0:window
        if i+l<=size
            if signal(i+l)>0
                if signal(i+l)-g*max>0
                    signal_cc(i+l)=signal(i+l)-g*max;
                else
                    signal_cc(i+l)=0;
                end
            else
                if signal(i+l)+g*max<0
                    signal_cc(i+l)=signal(i+l)+g*max;
                else
                    signal_cc(i+l)=0;
                end
            end
        end
    end

    i=i+window;
end

```

### A.3 center\_clipping.m (Variante Hüllkurve)

```

%Funktion für die Berechnung der Enevelope und danach Streckung des Signals
%Der Funktion muss der Signalvektor übergeben werden
%Fuktion ist nur innerhalb von p_track aufzurufen
%Aufruf: center_clipping('signalvektor');
%*****
function signal_cc=center_clipping(signal)
size=length(signal);
g=0.4;
abkling=0.999;

envelope(1)=signal(1);
signal_cc(1)=(1-g)*signal(1);

for i=2:size

    if signal(i)>=envelope(i-1)
        envelope(i)=signal(i);
    else
        envelope(i)=abkling*envelope(i-1);
    end

    if signal(i)>0

        if signal(i)-g*envelope(i)>0
            signal_cc(i)=signal(i)-g*envelope(i);
        else
            signal_cc(i)=0;
        end

    else

        if signal(i)+g*envelope(i)<0
            signal_cc(i)=signal(i)+g*envelope(i);
        else
            signal_cc(i)=0;
        end

    end

end

end

```

### A.4 leist.m

```

%Funktion zur Erkennung ob Signal voiced/unvoiced
%Der Funktion muss ein Filename übergeben werden
%Aufruf: leist('file');
%*****
function leist(file)

clear signal e_top e_bottom delta
[signal f_abtast]=wavread(file);

%Zeitskalierung
t=0:size(signal)-1;
t=t./f_abtast;

%Rauschen überlagern
%*****
%r_faktor=0.5;
%signal=signal + rand(size(signal),1).*r_faktor;

%Sinus überlagern
%*****
%f_sinus=17000;
%s_faktor=0.1;
%ueb_sinus=s_faktor.*sin(f_sinus*2*pi.*t)';
%signal=signal+ueb_sinus;

%Leistungsberechnung
leistung=signal.*signal;

%Filterung
f_lowpass=50;
[B,A]=butter(6,f_lowpass/(f_abtast/2));
leistunggefiltert=filtfilt(B,A,leistung);

%Bildung der Envelopen
min_signal=1/(128^2);
max_signal=max(leistunggefiltert);

```

```

tau=5;
e_top = zeros(size(signal),1);
e_bottom = zeros(size(signal),1);
quotient = zeros(size(signal),1);
e_top(1)=leistunggefiltert(1);
e_bottom(1)=min_signal;
quotient(1)=e_top(1)/e_bottom(1);
faktor_top=exp(log(min_signal/max_signal)/(tau*f_abtast));
faktor_bottom=exp(log(max_signal/min_signal)/(tau*f_abtast));

for n=2:size(leistunggefiltert)

    e_top(n)=max(leistunggefiltert(n),e_top(n-1)*faktor_top);

    if leistunggefiltert(n)<=min_signal
        e_bottom(n)=min_signal;
    else
        e_bottom(n)=min(leistunggefiltert(n),e_bottom(n-1)*faktor_bottom);
    end
end

%Quotient in dB
quotient=10.*log10(e_top./e_bottom);

%Ploten
figure
%set(gcf, 'toolbar', 'none')
%set(gcf, 'menubar', 'none')
subplot(311)
plot(t,signal)
axis([0 max(t) -1 1])
title (file)
subplot(312)
plot(t,leistung)
axis([0 max(t) 0 max(leistung)])
title ('Leistung')
subplot(313)
plot(t,leistunggefiltert)
axis([0 max(t) 0 max(leistunggefiltert)])
title ('Leistung gefiltert')
xlabel ('Time [s]')

figure
%set(gcf, 'toolbar', 'none')
%set(gcf, 'menubar', 'none')
subplot(311)
plot(t,signal)
axis([0 max(t) -1 1])
title(file)
subplot(312)
plot(t,leistunggefiltert,t,e_top,t,e_bottom)
axis([0 max(t) 0 max(leistunggefiltert)])
title ('Leistung&Envelopen')
subplot(313)
plot(t,quotient)
axis([0 max(t) 0 max(quotient)])
title ('Quotient')
xlabel ('Time [s]')
ylabel('[dB]')

```

## A.5 ausgabe.m

```

%Funktion zum Ausploten des mit Visual C++ erzeugten TXT-File
%Der Funktion muss der Filename.txt übergeben werden
%Aufruf: ausgabe('file.txt');
%*****
function ausgabe(file)
[Signal Detection Quotient RhoMax] = textread(file);

%Länge einlesen
SizeSignal = Signal(1)-1;
SizeRhoMax = RhoMax(1)-1;

%Zeitachsen
t = (0:1:SizeSignal);
t = t./22050;
tRho = (0:1:SizeRhoMax);
tRho = tRho.*(SizeSignal/(SizeRhoMax*22050));

%Daten einlesen
Signal = Signal(2:end);

```

```

Detection = Detection(2:end);
Quotient = Quotient(2:end);
RhoMax=RhoMax(2:SizeRhoMax+2);

%Ploten: Signal & Detection
figure
    set(gcf, 'toolbar', 'none')
    set(gcf, 'menubar', 'none')
    subplot(211)
    plot(t,Signal)
    axis([0 max(t) -1 1])
    title (file)

subplot(212)
    plot(t,Detection)
    axis([0 max(t) 0 1.2])
    title ('Detektion')
    xlabel ('Time [s]')

%Ploten: Quotient & RhoMax
figure
    set(gcf, 'toolbar', 'none')
    set(gcf, 'menubar', 'none')
subplot(211)
    Quotient = 10*log10(Quotient);
    plot(t,Quotient)
    ylabel('[dB]')
    title ('Quotient')
    axis([0 max(t) 1 max(Quotient)])
subplot(2,1,2)
    plot(tRho, RhoMax)
    axis([0 max(tRho) 0 1.2])
    title ('RhoMax')
    xlabel ('Time [s]')

```

## A.6 signalrausch.m

```

%Signalrauschabstand für totemuegerli berechnen
Signal1=wavread('totemuegerli1.wav');
Signal2=wavread('totemuegerli2.wav');
Signal3=wavread('totemuegerli3.wav');
Signal4=wavread('totemuegerli4.wav');
Signal5=wavread('totemuegerli5.wav');
S1 = 0;
S2 = 0;
S3 = 0;
S4 = 0;
S5 = 0;
for i = 1 : length(Signal1)
    S1 = S1 + Signal1(i).^2; %Signalleistung ohne Rauschen
    S2 = S2 + Signal2(i).^2;
    S3 = S3 + Signal3(i).^2;
    S4 = S4 + Signal4(i).^2;
    S5 = S5 + Signal5(i).^2;
end

SN2 = 10*log10(energy1/(energy2-energy1))
SN3 = 10*log10(energy1/(energy3-energy1))
SN4 = 10*log10(energy1/(energy4-energy1))
SN5 = 10*log10(energy1/(energy5-energy1))

```

## B Listing C++

- speech.h
- speech.cpp
- main.cpp
- speech.exe

### B.1 header.h

```
#ifndef __speech
#define __speech

#include <iostream.h>
#include <string.h>
#include <dsound.h>
#include <mmreg.h>

class speech {

long lSignalSize;//Anzahl Samples des Signals
char * cFileName;//File, welches bearbeitet wird
char * cSignal; //Samples Format char
char * cFile; //Eingelesener Filename
double * dSignal;//Samples Format double
double * dSignalPower;//Leistung Signal (x[n]^2)
double dMaxSignalPower;//Maximale Leistung
double * dEnvTop;//Obere Envelope der Leistung
double * dEnvBottom;//Untere Envelope der Leistung
double * dPowerQuotient;//Leistungsgr., voiced/unvoiced
double * dQuadSum;//P aufsummiert (sum(x[n]^2)
double * dRhoMax;//Max. der Autokor. im Fenster
double * dRhoMaxEnvel;//Envelope von RhoMax
int iSizeRhoMax; //Anzahl Werte des Vektors RhoMax
double * dPitch; //Pitchfrequenz
int * iDetection;//Vektor, stimmhaft/stimmlos
WAVEFORMATEX InfoWave;//Wave Informationen

public:

//Operationen
speech();
~speech();
bool OpenWaveFile(char *);
void WriteTxtFile(char *);
char *& Dialog();
void CharToFloat();
void AutoCorr();
void PitchCorrect();
void Power();
void Filter(int, double *, int);
void GetKoeff(int, double *, double * );
void Detection();
};

#endif __speech
```

## B.2 Operationen

### B.2.1 OpenWaveFile

```
//Daten in cSignal einlesen
bool speech::OpenWaveFile(char * FileName){

    ifstream hFile;
    char * csID = new char[5];
    csID[4] = '\0';
    long lFileSize;
    long lWaveFormatSize;

    hFile.open(FileName, ios::binary | ios::nocreate);

    if (!hFile){
        cerr << "Kann File nicht öffnen";
        return 1;
    }

    hFile.read(csID, 4);
    if (strcmp(csID,"RIFF")){
        cerr << "Hat keine RIFF ID";
        return 1;
    }

    hFile.read ((char*)&lFileSize, 4);

    hFile.read(csID, 4);
    if (strcmp(csID,"WAVE")){
        cerr << "Ist nicht vom Typ WAVE, sondern vom Typ " << csID;
        return 1;
    }

    hFile.read(csID, 4);
    if (strcmp(csID,"fmt ")){
        cerr << "Hat nicht die fmt chunk ID, sondern " << csID;
        return 1;
    }

    hFile.read((char*)&lWaveFormatSize, 4);

    hFile.read((char*) &InfoWave, lWaveFormatSize);

    if (InfoWave.wFormatTag != WAVE_FORMAT_PCM){
        cerr << "Wave Format ist nicht vom Typ PCM, sondern vom Typ " << InfoWave.wFormatTag;
        return 1;
    }

    hFile.read(csID, 4);
    if (strcmp(csID,"data")){
        cerr << "Die zweite chunk ID ist nicht data, sondern " << csID;
        return 1;
    }

    hFile.read((char*)&lSignalSize, 4);
    cSignal = new char [lSignalSize+1];
    cSignal[lSignalSize] = '\0';

    hFile.read(cSignal, lSignalSize);

    return 0;
}
```

### B.2.2 CharToFloat

```
//cSignal in dSignal konvertieren (char->double)
void speech::CharToFloat(){

    int k = 0;
    int iQuotient= 1;
    dSignal = new double[lSignalSize];
    dSignalPower= new double[lSignalSize];
    dQuadSum= new double[lSignalSize];

    iQuotient = pow(2,InfoWave.wBitsPerSample-1);
    while (k<lSignalSize) {
```

```

dSignal[k] = cSignal[k];

//Normierung |dSignal|<=1
dSignal[k]=dSignal[k]/iQuotient;

if (dSignal[k] < 0)
    dSignal[k] = dSignal[k] + 1;
else
    dSignal[k] = dSignal[k] - 1;

//Signal quadrieren, Vorarbeit für das Signalleistungsverfahren
dSignalPower[k] = dSignal[k] * dSignal[k];
    if (dSignalPower[k] > dMaxSignalPower)
        dMaxSignalPower = dSignalPower[k];

//Quadrieren und aufsummieren, Vorarbeit für die Autokorrelation
if(k==0)
    dQuadSum[k] = dSignalPower[k];
else
    dQuadSum[k]= dQuadSum[k-1] + dSignalPower[k];

k++;
}
}

```

## B.2.3 AutoCorr

```

//dSignal schrittweise autokorrolieren, maximale Autokorrelation -> dRhoMax.
//Bestimmung der Pitchfrequenz ->dPitch, keine weitere Verwendung für Detektion.
void speech::AutoCorr(){

int k=0; //Laufvariable für dRhoMax
int i=0; //Laufvariable für dRho
    int iVerschiebung=0;
    int iTmin = InfoWave.nSamplesPerSec/600;
    int iTmax = InfoWave.nSamplesPerSec/80;
    int iPhonem= 2*iTmax;
    int iTime = iTmax + iPhonem;
    int iStepWidth=5*InfoWave.nSamplesPerSec/1000;
    double dRho= 0;
    double dSqX= 0;
    double dSqY= 0;
    dRhoMax = new double[(lSignalSize-iTime)/iStepWidth];
    dRhoMaxEnvel=new double[(lSignalSize-iTime)/iStepWidth];
    double dUGrenze=1/1024.0;
    double dEnvelFaktor=exp(log(dUGrenze)/(5.0*(lSignalSize-iTime)/iStepWidth));
    iSizeRhoMax= 0;
    dPitch = new double[(lSignalSize-iTime)/iStepWidth];
    double dQuotient=0;

//Filter(800,dSignal,lSignalSize);

while(iTime <= lSignalSize) {

    dSqY = dQuadSum[iTime]-dQuadSum[iTime-iPhonem];
    dRhoMax[k] = 0;

    while(iTime-iTmin-iVerschiebung >= iTime-iTmax) {

        dSqX = dQuadSum[iTime-iTmin-iVerschiebung]-dQuadSum[iTime-iTmin-iVerschiebung-iPhonem];

        dQuotient = sqrt(dSqY*dSqX);

        if (dQuotient != 0){

            while(i<iPhonem) {
                dRho = dRho + (dSignal[i+iTime-iPhonem]*dSignal[i+iTime-iTmin-iVerschiebung-iPhonem])/
dQuotient;
                i++;
            }

            if(dRhoMax[k] < dRho){
                dRhoMax[k] = dRho;
                dPitch[k] = InfoWave.nSamplesPerSec/(iVerschiebung + iTmin);
            }

        }

    }

}
}

```

```

        else{
        }

        iVerschiebung++;
        i = 0;
        dRho = 0;

    }

    if (k!=0){
        dRhoMax[k]=max(dRhoMax[k-1]*dEnvelFaktor,dRhoMax[k]);
    }

    iSizeRhoMax++;
    k++;
    iVerschiebung = 0;
    iTime = iTime + iStepWidth;
}
}

```

## B.2.4 Power

```

//Quadrierte Signal filtern, obere und untere Enveloppe -> dPowerQuotient
void speech::Power(){

    Filter(50, dSignalPower, lSignalSize);

    float delta = 0;

    dEnvTop      = new double[lSignalSize];
    dEnvBottom= new double[lSignalSize];
    dPowerQuotient= new double[lSignalSize];

    double dMinSignalPower= 1/pow(pow(2,InfoWave.wBitsPerSample-1),2);
    double dTau  = 5;
    //Quotient Annahme: 5000 Maximum
    double dEnvFaktor= exp(log(1/5000)/(dTau*InfoWave.nSamplesPerSec));

    dEnvTop[0]= dSignalPower[0];
    dEnvBottom[0]= dMinSignalPower;
    dPowerQuotient[0]= dEnvTop[0]/dEnvBottom[0];

    double dFaktorTop= exp(log(dMinSignalPower/dMaxSignalPower)/(dTau * InfoWave.nSamplesPerSec));
    double dFaktorBottom= 1 + (1 - dFaktorTop);

    for(int i = 1; i < lSignalSize; i++){

        dEnvTop[i] = max(dSignalPower[i], dEnvTop[i-1] * dFaktorTop);

        if (dSignalPower[i] <= dMinSignalPower){
            dEnvBottom[i] = dMinSignalPower;
        }
        else{
            dEnvBottom[i]=min(dSignalPower[i], dEnvBottom[i-1] * dFaktorBottom);
        }

        dPowerQuotient[i] = dEnvTop[i]/dEnvBottom[i];
        //dPowerQuotient glätten
        dPowerQuotient[i] = max(dPowerQuotient[i], dPowerQuotient[i-1]*dEnvFaktor);

    }
}

```

## B.2.5 Detection

```

//Entscheid fällen ob stimmhaft (0/1)
void speech::Detection(){
    int iTmax=InfoWave.nSamplesPerSec/80;
    int iStep=5*InfoWave.nSamplesPerSec/1000;
    int k = 0;
    int i = 0;
    double * dRhoMaxTemp;
    dRhoMaxTemp = new double [lSignalSize];
    iDetection = new int [lSignalSize];

    //RhoMax strecken auf lSignalSize

```

```

while(i<iSizeRhoMax){
    if (i==0){
        for(int s = 0; s < 3*iTmax; s++){
            dRhoMaxTemp[k] = dRhoMax[i];
            k++;
        }
    }
    else{
        for(int s = 0; s < iStep; s++){
            dRhoMaxTemp[k] = dRhoMax[i];
            k++;
        }
    }
    i++;
}

for(int s = k; s < lSignalSize; s++){
    dRhoMaxTemp[s] = dRhoMax[i-1];
}

i = 0;

//Abfrage
while(i<lSignalSize){
    if(i < 1.5*InfoWave.nSamplesPerSec){
        if(dRhoMaxTemp[i]>0.8){
            iDetection[i] = 1;
        }
        else{
            iDetection[i] = 0;
        }
    }
    else{
        if((dRhoMaxTemp[i]>0.8) && (dPowerQuotient[i]>30)){
            iDetection[i] = 1;
        }
        else{
            iDetection[i] = 0;
        }
    }
    i++;
}
}

```

## B.2.6 Filter

```

//Filterung möglich für Abtastfrequenz:22050Hz
// für Grenzfrequenz: 50, 75, 100, 150, 600, 800 Hz
void speech::Filter(int GrenzFreq, double * x,int iSize){
    int i;
    double a[7];
    double b[7];

    GetKoeff(GrenzFreq, a, b);

    double * y;
    y = new double [iSize];

    for (i = 0; i < iSize ; i++){
        switch (i)
        {
            case 0 :
            {
                y[i]=(b[0]*x[i])/a[0];
                break;
            }
            case 1 :
            {
                y[i]=(b[0]*x[i]+b[1]*x[i-1]
                    -a[1]*y[i-1])/a[0];
                break;
            }
            case 2 :
            {
                y[i]=(b[0]*x[i]+b[1]*x[i-1]+b[2]*x[i-2]
                    -a[1]*y[i-1]-a[2]*y[i-2])/a[0];
                break;
            }
        }
    }
}

```

```

case 3 :
{
y[i]=(b[0]*x[i]+b[1]*x[i-1]+b[2]*x[i-2]+b[3]*x[i-3]
-a[1]*y[i-1]-a[2]*y[i-2]-a[3]*y[i-3])/a[0];
break;
}
case 4 :
{
y[i]=(b[0]*x[i]+b[1]*x[i-1]+b[2]*x[i-2]+b[3]*x[i-3]+b[4]*x[i-4]
-a[1]*y[i-1]-a[2]*y[i-2]-a[3]*y[i-3]-a[4]*y[i-4])/a[0];
break;
}
case 5 :
{
y[i]=(b[0]*x[i]+b[1]*x[i-1]+b[2]*x[i-2]+b[3]*x[i-3]+b[4]*x[i-4]+b[5]*x[i-5]
-a[1]*y[i-1]-a[2]*y[i-2]-a[3]*y[i-3]-a[4]*y[i-4]-a[5]*y[i-5])/a[0];
break;
}
default :
{
y[i]=(b[0]*x[i]+b[1]*x[i-1]+b[2]*x[i-2]+b[3]*x[i-3]+b[4]*x[i-4]+b[5]*x[i-5]+b[6]*x[i-6]
-a[1]*y[i-1]-a[2]*y[i-2]-a[3]*y[i-3]-a[4]*y[i-4]-a[5]*y[i-5]-a[6]*y[i-6])/a[0];
}
}
}
for (i = 0; i < iSize ; i++){
x[i] = y[i];
}
}

```

## B.2.7 GetKoeff

```

//Koeffizienten in Matlab berechnet
void speech::GetKoeff(int iGrenzFreq, double * a, double * b){
int AnzKoeff = 7;
int k = 0;
double B[6][7] = {
// Grenzfrequenz 50 Hz, Abtastfreq 22050
0.01271743127473e-011,0.0763045876484e-011,0.190761469121e-011,0.25434862549467e-
011,0.190761469121e-011,0.0763045876484e-011,0.01271743127473e-011,
// Grenzfrequenz 75 Hz, Abtastfreq 22050
0.01428818868776e-010,0.08572913212657e-010,0.21432283031642e-010,0.28576377375522e-
010,0.21432283031642e-010,0.08572913212657e-010,0.01428818868776e-010,
//Grenzfrequenz 100 Hz, Abtastfreq 22050
0.00791982227837e-009,0.04751893670218e-009,0.11879734175546e-009,0.15839645567395e-
009,0.11879734175546e-009,0.04751893670218e-009,0.00791982227837e-009,
//Grenzfrequenz 150 Hz, Abtastfreq 22050
0.00878067358406e-008,0.05268404150438e-008,0.13171010376095e-008,0.17561347168127e-
008,0.13171010376095e-008,0.05268404150438e-008,0.00878067358406e-008,
//Grenzfrequenz 600 Hz, Abtastfreq 22050
0.02844775765284e-005,0.17068654591702e-005,0.42671636479256e-005,0.56895515305674e-
005,0.42671636479256e-005,0.17068654591702e-005,0.02844775765284e-005,
//Grenzfrequenz 800 Hz, Abtastfreq 22050
0.01447175531117e-004,0.08683053186704e-004,0.21707632966760e-004,0.28943510622347e-
004,0.2170763296676e-004,0.08683053186704e-004,0.01447175531117e-004};

double A[6][7] = {
// Grenzfrequenz 50 Hz, Abtastfreq 22050
1.0,-5.94495164724660,14.72627135177846,-19.455542680006,14.45851649060017,-
5.73073232583183,0.94643873871394,
// Grenzfrequenz 75 Hz, Abtastfreq 22050
1.0,-5.91742764208184,14.59054040579927,-19.18779653950875,14.19442514053908,-
5.6004840585337,0.92074269387739,
//Grenzfrequenz 100 Hz, Abtastfreq 22050
1.0,-5.88990384231897,14.455563434519,-18.92300559990598,13.9346794482777,-
5.4730765530755,0.89574311301063,
//Grenzfrequenz 150 Hz, Abtastfreq 22050
1.0,-5.83485713244348,14.18786654029455,-18.40219007384664,13.42796654743177,-
5.22654284752026,0.84775697170368,
//Grenzfrequenz 600 Hz, Abtastfreq 22050
1.0,-5.33957525339630,11.91243469795633,-14.21060176376399,9.55867038939104,-
3.43693295276875,0.51602308914657,
//Grenzfrequenz 800 Hz, Abtastfreq 22050
1.0,-5.11959380577607,10.97719977465459,-12.61179830801334,8.18557727309767,-
2.84474884043065,0.41345652570179};

if (iGrenzFreq == 50 && InfoWave.nSamplesPerSec == 22050)
k = 0;
else

```

```

if (iGrenzFreq == 75 && InfoWave.nSamplesPerSec == 22050)
k = 1;
else
if (iGrenzFreq == 100 && InfoWave.nSamplesPerSec == 22050)
k = 2;
else
if (iGrenzFreq == 150 && InfoWave.nSamplesPerSec == 22050)
k = 3;
else
if (iGrenzFreq == 600 && InfoWave.nSamplesPerSec == 22050)
k = 4;
else
if (iGrenzFreq == 800 && InfoWave.nSamplesPerSec == 22050)
k = 5;
else
cout << "Es existieren keine Filterkoeffizienten zu dieser Grenzfrequenz und Ab-
tastfrequenz!!!" << endl;

for(int i = 0; i < AnzKoeff; i++){
a[i] = A[k][i];
b[i] = B[k][i];
}
}

```

## B.2.8 Dialog

```

//GUI, vor dem Ausführen ev. Pfad ändern
char *& speech::Dialog(){

char * cFileName= new char[100];
cFile = new char[100];
char * cFolderAudio= "d:/semesterarbeit/audio/";
char * cFolderAudioNotSpeech= "d:/semesterarbeit/audio/notspeech/";
char * cFolderAudioSpeech= "d:/semesterarbeit/audio/speech/";
char * cWave = ".wav";
ifstream hFile;
ifstream hFile1;

cout << "Geben Sie bitte den Filenamen ein: " << endl;

cin >> cFile;

hFile1.open(cFile, ios::nocreate);

if(hFile1){
hFile1.close();
cout << "Pfad fuer Wavefile: " << cFile<< endl;
return cFile;
}

strcpy(cFileName,cFolderAudio);
strcat(cFileName,cFile);
strcat(cFileName,cWave);

hFile.open(cFileName, ios::nocreate);

if (hFile){
hFile.close();
cout << "Pfad fuer Wavefile: " << cFileName << endl;
return cFileName;
}
else {
delete cFileName;
delete hFile;
ifstream hFile;
char * cFileName = new char[100];
strcpy(cFileName,cFolderAudioSpeech);
strcat(cFileName,cFile);
strcat(cFileName,cWave);
hFile.open(cFileName,ios::nocreate);
if (hFile){
hFile.close();
cout << "Pfad fuer Wavefile: " << cFileName << endl;
return cFileName;
}
else {
delete cFileName;
int k = 10;
delete hFile;
ifstream hFile;
char * cFileName = new char[100];

```

```

strcpy(cFileName,cFolderAudioNotSpeech);
strcat(cFileName,cFile);
strcat(cFileName,cWave);
hFile.open(cFileName,ios::nocreate);
if (hFile){
    hFile.close();
    cout << "Pfad fuer Wavefile: " << cFileName << endl;
    return cFileName;
}
else {
    cerr << "Sie haben ein falschen Dateinamen oder Pfad eingegeben! " << endl << endl;
    char * cFehler = "FEHLER";
    return cFehler;
}
}
}
}
}

```

## B.2.9 WriteTxtFile

```

//Daten ins Textfile schreiben
void speech::WriteTxtFile(char * cTxtName) {

    int  ilaenge = strlen(cTxtName);
    char * cTxt= "txt";
    ofstream TxtFile;

    strcpy(cTxtName+(ilaenge-3),cTxt);

    cout << "Pfad fuer Textfile: " << cTxtName << endl;

    TxtFile.open(cTxtName, ios::out | ios::trunc);

    if(!TxtFile)
        cerr << "Fehler beim Oeffnen der Datei " << cTxtName << "!" << endl;

    TxtFile << lSignalSize << '\t' << lSignalSize << '\t' << lSignalSize << '\t' << iSizeRhoMax /* << '\t'
<< lSignalSize << '\t' << lSignalSize */ << endl;

    for(int i = 0; i < lSignalSize; i++) {

        if(i < iSizeRhoMax)
            TxtFile << dSignal[i] << '\t' << iDetection[i] << '\t' << dPowerQuotient[i] << '\t' << dRhoMax[i]
/*<< '\t' << dSignalPower[i] << '\t' << dEnvTop[i] << '\t' << dEnvBottom[i]*/ << endl;
        else
            TxtFile << dSignal[i] << '\t' << iDetection[i] << '\t' << dPowerQuotient[i] /* << '\t' << 0 <<
'\t' << dSignalPower[i] << '\t' << dEnvTop[i] << '\t' << dEnvBottom[i] */<< endl;
    }

    TxtFile.close();
}
}

```