Sound acoustics Research, Development, Implementation

HASQUE
-PCM Hörtestsimulation-

HASQUE DLL VERSION 8.8

FÜR WINDOWS BETRIEBSSYSTEME

ZUR OBJEKTIVEN QUALITÄTSBEURTEILUNG

VON AUDIOSYSTEMEN

Edition:	**1.3 – 15.02.2021**
HASQUE Softwarestand:	**V 8.8**
Datum:	**04.03.2021**

Inhalt

Abkürzungen

CTR	CochleaTransformation
dBov	Pegel unter Vollaussteuerung
DLL	Dynamic Link Library
GAL	Gain Alignment
MOS	Mean Opinion Score
SAM	Short Average Magnitude
SPE	Signed Perceptible Error
SPL	Sound pressure level
TAL	Time Alignment

Lieferumfang

HASQUEDll.lib	Importbibliothek
HASQUEDll.dll	Dynamische Bibliothek
HASQUEDLL.h	Header mit Strukturen und Funktionen der DLL
HASQUEDllApi.h	Header der Klasse HASQUEDllApi
HASQUEDllApi.cpp	Klasse HASQUEDllApi DLL Anwendung
HASQUEDLLHelp.pdf	Beschreibung der DLL mit Anleitung zur Anwendung

Nutzungsvereinbarungen

Die Verwendung der im Lieferumfang aufgeführten Bestandteile und die Nutzung dieser Software und/oder der dazugehörenden Dokumentation, Quellcodes, Bibliotheken derselben ist nur in Verbindung mit einer vertraglichen Vereinbarung mit Sound acoustics oder dessen Beauftragte, nur nach erfolgter Lizenzzahlung und nur unter der Voraussetzung erlaubt, dass folgende Regeln vom Nutzer akzeptiert und eingehalten werden:

- Die DLL beinhaltet Lösungen, deren Bestandteile geistiges Eigentum von Sound acoustics sind und nicht zu einem anderen Zweck als vorgesehen verwendet werden dürfen.
- Jegliche Nachahmung, Dekompilierung oder Reverse Engineering ist verboten.
- Jegliche Änderung der gelieferten DLL ist unzulässig.
- Eine kommerzielle Nutzung ist nur mit den von Sound acoustics erhältlichen Dauerlizenzen erlaubt.
- Eine kommerzielle Nutzung unter Inanspruchnahme der Entwicklerlizenz ist nicht gestattet.
- Jegliche Weitergabe der im Lieferumfang aufgeführten Bestandteile an Dritte, die nicht durch diesen Vertrag mit einer an Sound acoustics zu entrichtenden Lizenzgebühr abgedeckt ist, ist unzulässig.
- Das Nutzungsrecht gilt jeweils für einen PC (keine Multiuserlizenz) und ist nicht übertragbar.
- HAFTUNGSAUSSCHLUSS: Sound acoustics übernimmt keinerlei Haftung für etwaige Schäden die direkt oder indirekt auf die Verwendung der DLL oder anderen von Sound acoustics bereitgestellten Software- und Hardwarekomponenten zurückgeführt werden könnten.

- Sonstige Haftungen, die nicht unter den Haftungsausschluss fallen sind auf einmalig 10 Euro begrenzt.

Funktionen und Schnittstellen der HASQUEDllApi

Die DLL beinhaltet den HASQUE Hörtestsimulator (**H**earing **A**dequate **S**ignal **Qu**ality **E**valuation) zur Qualitätsbeurteilung von Audio- und Telekommunikationssystemen und zusätzliche Messverfahren zur Ermittlung der Signaleigenschaften.

Die HASQUEDllApi ermöglicht eine einfache Implementierung und Anwendung der DLL Funktionen in einem Anwenderprogramm. Die Schnittstellen der DLL sind als Pointer zu Speicherplätzen des Anwenderprogramms ausgelegt. Die Klasse legt notwendige Speicherplätze an und initialisiert die DLL mit einer Qualitätsskala nach ITU-T P.862 und Hörtestparametern nach Anforderungen der BDBOS.

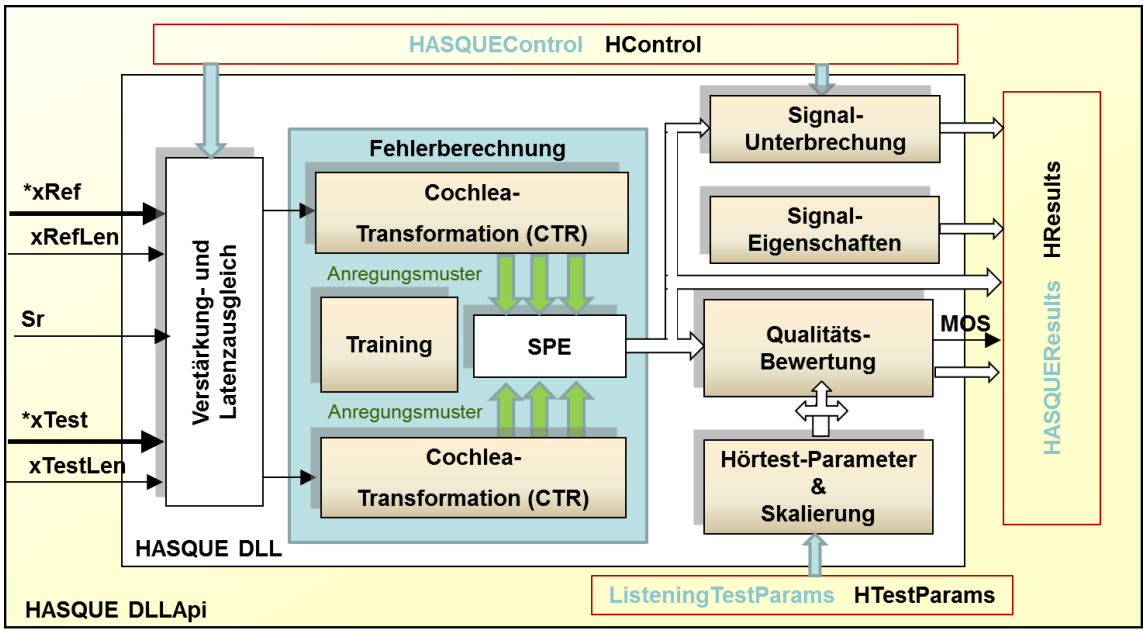

Bild 1: HASQUEDLLApi Schnittstellen und Funktionen

Bild 1 zeigt die Schnittstellen und funktionalen Zusammenhänge der HASQUEDLLApi. Mit dem Anlegen eines Objekts dieser Klasse in einem Anwenderprogramm werden alle DLL Funktionen verfügbar gemacht und vorinitialisiert.

Fehlerberechnung

Mit dem HASQUE Verfahren werden die menschlichen Höreigenschaften über eine sogenannte Cochleatransformation (CTR) ohne Bandbegrenzung nachgebildet. Die CTR bildet hierbei in Abhängigkeit der Hörtestparameter verschiedene Anregungsmuster nach, welche mit der psychoakustischen Wahrnehmung des menschlichen Gehörs verglichen werden können.
Die in der CTR nachgebildete Mithörschwelle wird an das Hintergrundgeräusch der eingespeisten Signale in Abhängigkeit der Geräuscheigenschaften adaptiert. So werden quasi stationäre Geräuschquellen, wie beispielsweise Fahrgeräusche bei konstanter Geschwindigkeit und nicht stationäre Geräusche, wie das

Knattern eines vorbeifahrenden Zuges unterschiedlich stark bewertet. Gewöhnungseffekte (Training) und Hintergrundgeräuscheigenschaften werden bei der Bewertung somit berücksichtigt, wodurch eine hohe Korrelation mit der subjektiv empfundenen Qualität erreicht werden kann.

Der Vergleich zwischen den über die CTR „gehörten“ Referenz- und Testsignalen ergibt die „hörbaren“ Fehler welche als SPE_Error (Signed Perceptible Error) bezeichnet sind. SPE gibt vorzeichenbehaftete Lautheitswerte der Störungen in Sone an. Negative Werte entsprechen Signalverluste, die ab einer bestimmten Größe auch als Signalunterbrechung aufgefasst werden können. Positive SPE_ERROR Werte entsprechen Signalüberhöhungen, die ab einer bestimmten Schwelle als Fremdsignal interpretiert werden können.

Hörtestparameter und Skalierung

HASQUE kann mit Hilfe der Hörtestparameter (Akzeptanzschwelle, Bandbreite und Abhörlautstärke) an verschiedene Hörtestbedingungen angepasst und mit individueller Qualitätsskala programmiert werden. Hörtestparameter werden nach Anforderungen der BDBOS und die Qualitätsskala nach ITU-T P.862 beim Anlegen eines Objekts der Klasse HASQUEDLLApi über den Konstruktor vorinitialisiert.

Bezeichnung	Format	Bedeutung	Default
SR	int	Abtastrate	8000
ThresholdOfAcceptance	float	Akzeptanzschwelle der Störungen in Sone	3.2
UpperFC	float	Obere Grenzfrequenz - ergibt sich aus SR	4000
LowerFC	float	Untere Grenzfrequenz	100
SystemLevel	float	Systempegel der Abhörlautstärke in dB(SPL)	-13
MOSmax	float	Obergrenze der MOS Skala (excellent)	4.5
MOSmin	float	Untergrenze der MOS Skala (bad)	-0.5
Compressed	bool	true = komprimierte sonst natürliche Fehlerbewertung	true

Einstellungen 1 : Hörtestparameter und Skalierung mit HTestParams

Eine Änderung der Hörtestparameter kann bei Bedarf mit Hilfe der als ListeningTestParams definierten Variable HTestParams zur Anpassung an andere Hörtestbedingungen durchgeführt werden. Die Qualitätsskala kann individuell festgelegt werden So kann beispielsweise eine prozentuale Angabe der Qualität im Bereich 0-100 oder nach ITU-T P.862.1 im Bereich 1-5 festgelegt werden.

Steuerung der DLL

Die Steuerung der DLL erfolgt mit Hilfe der als HASQUEControl definierten Variable HControl. Über diese Steuerung können sowohl Funktionen aktiviert bzw. deaktiviert werden, als auch Schwellenwerte und andere Parameter wie folgt festgelegt werden.

Verstärkungs- und Latenzausgleich

Für eine korrekte Bewertung müssen Referenz- und Testsignal zeitgleich und mit gleicher Amplitude „gehört“ werden. Die dafür verfügbaren Funktionen Time Alignment (TAL) und Gain Alignment (GAL) sind programmierbar und werden in der HASQUEDLLApi mit dem Konstruktor vorinitialisiert.

Bezeichnung	Format	Bedeutung	Default
isGAL	bool	true = GAL ein, sonst fester Verstärkungsfaktor	True
GainCorrDeg	float	Verstärkungsfaktor zur Amplitudenanpassung der Testsignale	1
isTAL	bool	true = TAL ein, sonst feste Verzögerung	True
Delay_p2	int	Verzögerungswert in Anzahl Abtastwerte feste Verzögerung	0
TALMax	float	Maximale Verzögerung in Sekunden	1
TALMin	float	Minimale Verzögerung in Sekunden	-0.2
isBlockDComp	bool	true = Blockkompensation ein sonst aus	True
isJitterDComp	bool	true = Latenzjitterkompensation ein sonst aus	False

Einstellungen 2: Time- und Gain Alignment über HControl

GAL kann für den Fall deaktiviert werden, wenn Qualitätsmessungen an Verfahren mit konstanter Verstärkung durchgeführt werden sollen. Dieser Sonderfall betrifft insbesondere die Parametrierung und Qualitätsoptimierung von Audiosystemen im Bereich Forschung und Entwicklung von neuen Verfahren. Bei isGAL = false muss GainCorrDeg auf den Kehrwert des Verstärkungsfaktors des zu bewertenden Verfahrens gesetzt werden.

Mit der Aktivierung der GAL-Funktion (isGAL = true) werden Lautheitsunterschiede zwischen Referenz- und Testsignal automatisch kompensiert. Der berechnete Verstärkungsfaktor wird mit HResults.GAL angegeben.

Bei Qualitätsmessungen von Audiosystemen mit bekannter und konstanter Latenz ist eine feste Vorgabe (isTAL = false, Delay_p2 = Vorgabe) empfehlenswert. Dies betrifft beispielsweise Messungen an Signalverarbeitungsverfahren wie Geräuschreduktionsverfahren, Bandbreitenerweiterung und Andere im Bereich Forschung und Entwicklung.

Mit der Aktivierung der TAL-Funktion (isTAL = true) wird eine konstante Laufzeitdifferenz zwischen Referenz- und Testsignal mit hoher Genauigkeit (typ. <1ms) ausgeglichen. Die gemessene Verzögerung wird mit HResults.TAL in Abtastwerten angegeben.

Bei aktiver TAL-Funktion kann eine zusätzliche Blockkompensation mit isBlockDComp =true aktiviert werden. Die Blockkompensation wird dann benötigt, wenn die Latenz innerhalb einer Aufnahme zwischen Referenz- und Testsignal durch eine zusätzliche Laufzeitschwankung variieren kann. Dies kann beispielsweise bei Funkübertragungssystemen bei Zellwechsel auftreten.

Alternativ zur Blockkompensation kann die Latenzjitterkompensation mit isJitterDComp=true aktiviert werden. Diese wird benötigt, wenn die Latenz innerhalb einer Aufnahme zwischen Referenz- und

Testsignal durch Laufzeitjitter schwanken kann. Laufzeitjitter kann bei einer Signalübertragung über IP Netze auftreten und erfordert die zeitliche Korrektur jeder einzelnen Sprachäußerung.

Eine Anpassung der Signallaufzeitgrenzen mit TALMAX und TALMIN ist beispielsweise dann notwendig, wenn Signallaufzeitunterschiede zwischen Referenz- und Testsignal mit der Vorinitialisierung nicht abgedeckt sind oder so gering sind, dass eine Einschränkung des Bereichs wegen der damit reduzierten Rechenleistung und erhöhten Robustheit gegen Störungen angezeigt ist.

Fehlerklassifizierung

Signalunterbrechungen können durch starke Signaldämpfung oder durch Störsignale (Artefakte) hervorgerufen werden. In beiden Fällen ist das Signal nicht mehr verständlich. Mit Hilfe von Schwellenwerten und Zeitangaben lassen sich die Funktionen zur Ermittlung individueller Fehlereigenschaften festlegen.

Name	Format	Meaning	Default
SPIRLOUDTHR1	float	Lautheitsschwelle (Sone)	10
SPIRMinIRTime1	float	minimale Anregungszeit (ms)	10
SPIRLOUDTHR2	float	wird für DLL nicht verwendet	2.5
SPIRMinIRTime2	float	wird für DLL nicht verwendet	0
ThreshOfDistSone	float	max. Lautheit der Störung (Sone)	15.75
ArtIntervall	float	Interval time of the distortion ms	100
AddIRTimes	bool	Bewerte zuzüglich Unterbrechungen	true
ThreshOfAttSone	float	maximale Auslöschung(Sone)	-45
ArtMOSThres	float	Max. mögliche Qualität	2.5
ArtSpecProperty	float	Spektrale Zusammensetzung 0 = Breitband- SNR Schmalb	0
ArtSpecF1	float	Grundfrequenz 1	0
ArtSpecF2	float	Grundfrequenz 2	0

Einstellungen 3: Signalunterbrechungen und programmierbare Fehler mit HControl

Sprachunterbrechung

Die Funktion zur Ermittlung der Sprachunterbrechungszeit innerhalb einer Aufnahme vergleicht den Betragswert von SPE_ERROR während der Sprachaktivität des Referenzsignals mit dem Schwellenwert SPIRLOUDTHR1 und ermittelt die Zeitdauer der Schwellwertüberschreitung. Ist die Zeitdauer kleiner als SPIRMinIRTime1, wird die Überschreitung nicht als Sprachunterbrechung gewertet, um zu verhindern, dass beispielsweise eine kurzzeitige Knackstörung als Sprachunterbrechung interpretiert wird.

Die Steuervariablen SPIRLOUDTHR1 und SPIRMinIRTime1 sind für „Standardmessungen“ vorinitialisiert und können bei Bedarf an andere Applikationen angepasst werden.

Programmierbare Fehlererkennung (Error Tracer)

Das Prinzip der Fehlererkennung basiert auf eine Nachbildung der neuronalen Verarbeitung und nutzt die statistische (Un-)Wahrscheinlichkeit aus, die sich bei der Kombination der Fehlereigenschaften eines speziellen Fehlers zwangsläufig ergeben. Das Verfahren kombiniert die Fehlereigenschaften mit individueller Parametrierung zu einem „Fingerabdruck“ und erreicht in der Regel eine Erkennungsrate von weit über 90%.

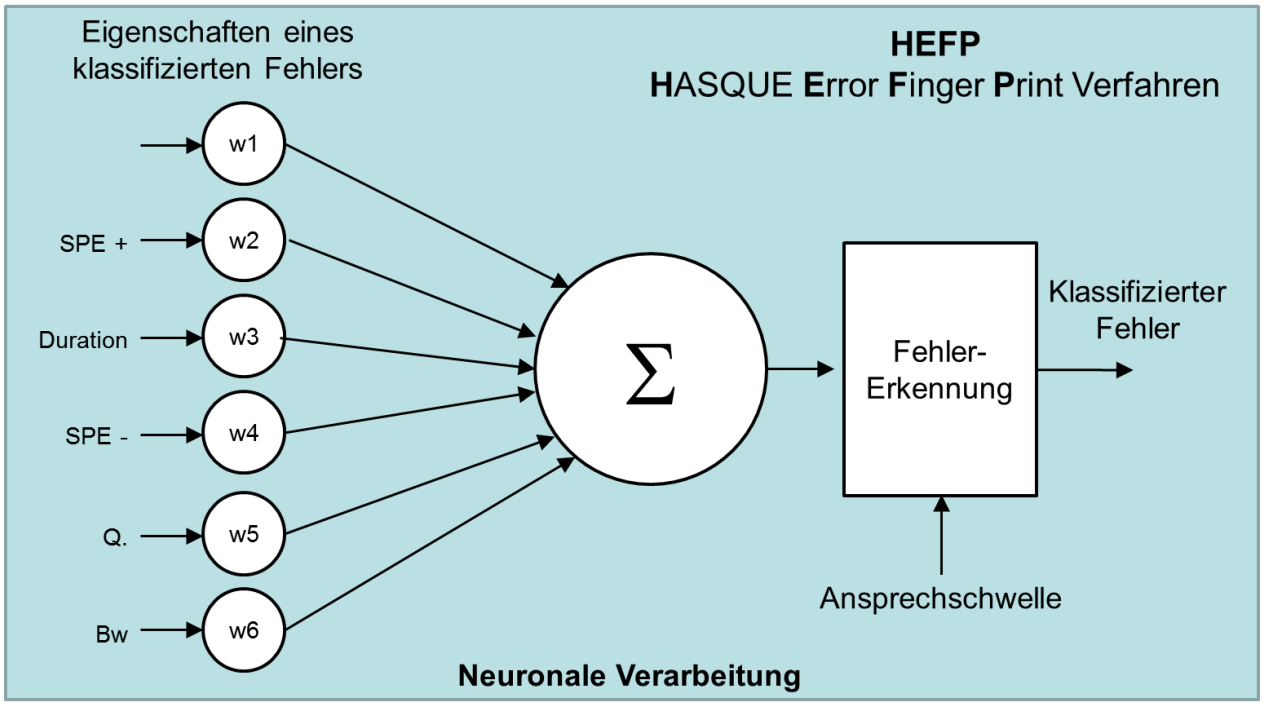

Bild 2: Fehlererkennung durch Merkmalssynthese

Die Fehlereigenschaften eines klassifizierten Fehlers können mit Hilfe der Einstellungen in HControl frei programmiert werden. Die Werte dieser Eigenschaften lassen sich mit Hilfe einer der HASQUE Messsysteme durch Einscannen der Fehler automatisch ermitteln.

Im Folgenden werden ein paar Beispiele für klassifizierte Fehler gezeigt.

Zellwechsel

Zellwechsel können bei einer Funkverbindung während einer Fahrt erzeugt werden, wenn das Funkgerät zwischen zwei Sendemasten umschaltet. Hierbei werden Artefakte erzeugt welche subjektiv als stark schwankende Ereignisse empfunden werden. Dieser Fehler wurde als CellReselection Fehler klassifiziert .

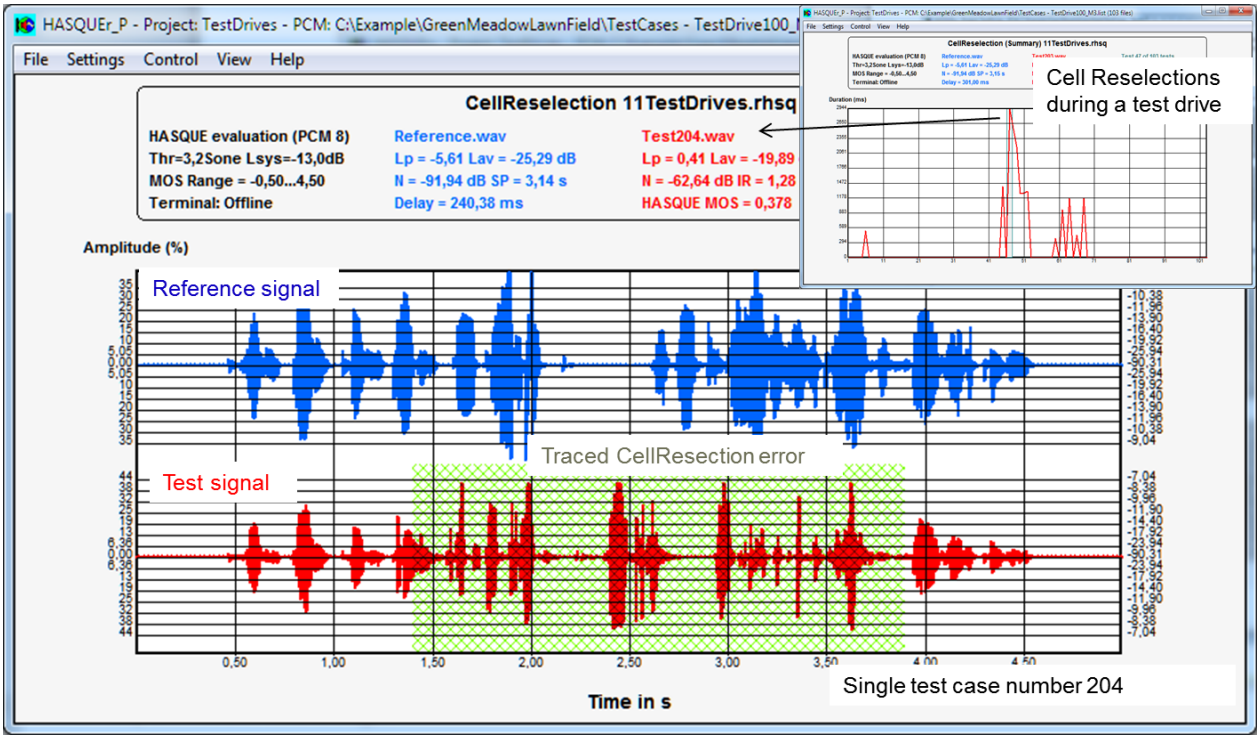

Bild 3: Messung von Zellwechselstörungen

Im Bild 3 oben rechts werden alle Testfälle einer Testfahrt angezeigt, bei denen Zellwechselstörungen auftraten. Die Größe zeigt hierbei die Zeit der Störung in ms an. Die Funktion zur Erkennung von individuellen Fehlern innerhalb einer Aufnahme stellt den Vektor ClassErr zur Verfügung und zeigt mit „Traced CellReselection error“ auf der Zeitachse an, wann der klassifizierte Fehler in den Aufnahmen vorkommt. Die Eigenschaften von Zellwechselstörungen sind wie folgt parametriert:

Signal Eigenschaften	Wert	Einheit
HControl.ThreshOfDistSone	15.748	Sone
HControl.ArtIntervall	100	ms
HControl.AddIRTimes	true	Bool
HControl.ThreshOfAttSone	-45	Sone
HControl.MaxCorrelation	79.927	%
HControl.ArtMOSThres	2.64	MOS
HControl.ArtSpecProperty	0	Factor
HControl.ArtSpecF1	0	Hz
HControl.ArtSpecF2	0	Hz

Einstellungen 4: Eigenschaften von Zellwechselstörungen

Obige Parameter wurden mit Hilfe eines Wizards des HASQUE Messsystems automatisch ermittelt und werden in der DLL per Default gesetzt.

Martinshorn (PoliceSiren)

Sowohl beim Freisprechen, als auch beim reinen Hörerbetrieb können dem Nutzsignal akustische Störungen durch das Martinshorn eines Einsatzfahrzeugs überlagern. Störungen dieser Art wurden im Folgenden als Martinshorn Störungen klassifiziert.

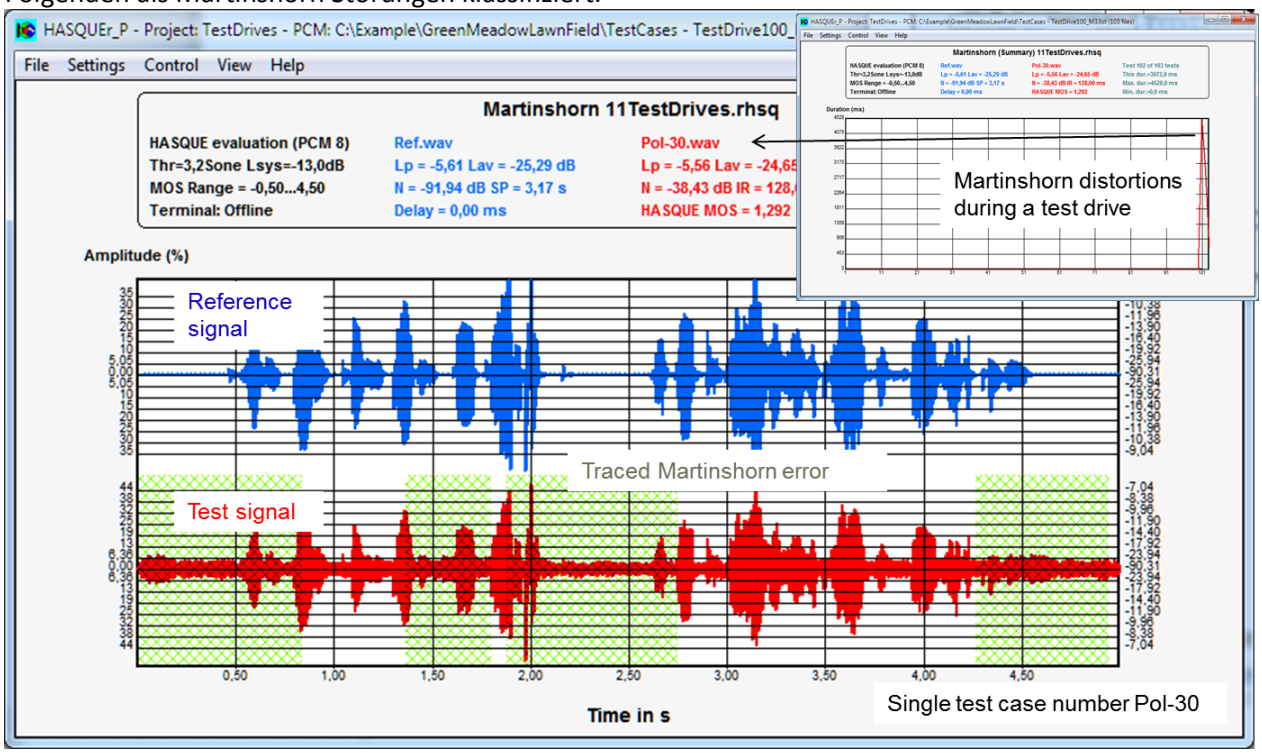

Bild 4: Störungen durch Martinshorn

Im kleinen Bild oben rechts in Bild 4 sind alle Fälle mit Störungen durch ein Martinshorn aufgezeigt, wobei Störungen durch ein Martinshorn erst am Ende der Testfahrt zu erkennen sind. Die Amplitude der roten Kurve zeigt die Zeitdauer der gesuchten Störung in jeder Aufnahme. Das übrige Bild zeigt die Darstellung eines ausgewählten Testfalles aus dieser Übersichtsgraphik mit Störungen durch ein Martinshorn als „Traced Martinshorn error“.
Folgende Einstellungen werden zum Aufspüren von Signalstörungen durch das Martinshorn benötigt:

Signal Eigenschaften	Wert	Einheit
HControl.ThreshOfDistSone	13.835	Sone
HControl.ArtIntervall	100	ms
HControl.AddIRTimes	true	Bool
HControl.ThreshOfAttSone	-0.7767	Sone
HControl.MaxCorrelation	91.883	%
HControl.ArtMOSThres	2.76	MOS
HControl.ArtSpecProperty	53.82	Factor
HControl.ArtSpecF1	453	Hz
HControl.ArtSpecF2	609	Hz

Einstellungen 5: Signaleigenschaften von Martinshorn - Störungen

Die Einstellungen müssen vor der Initialisierung und Evaluierung vorgenommen werden (Steuerung der Software).

FunkHoles

Hörbare Signalunterbrechungen können währen einer Testfahrt durch Funklöcher entstehen. Dieser Fehler wurde daher als FunkHoles klassifiziert.

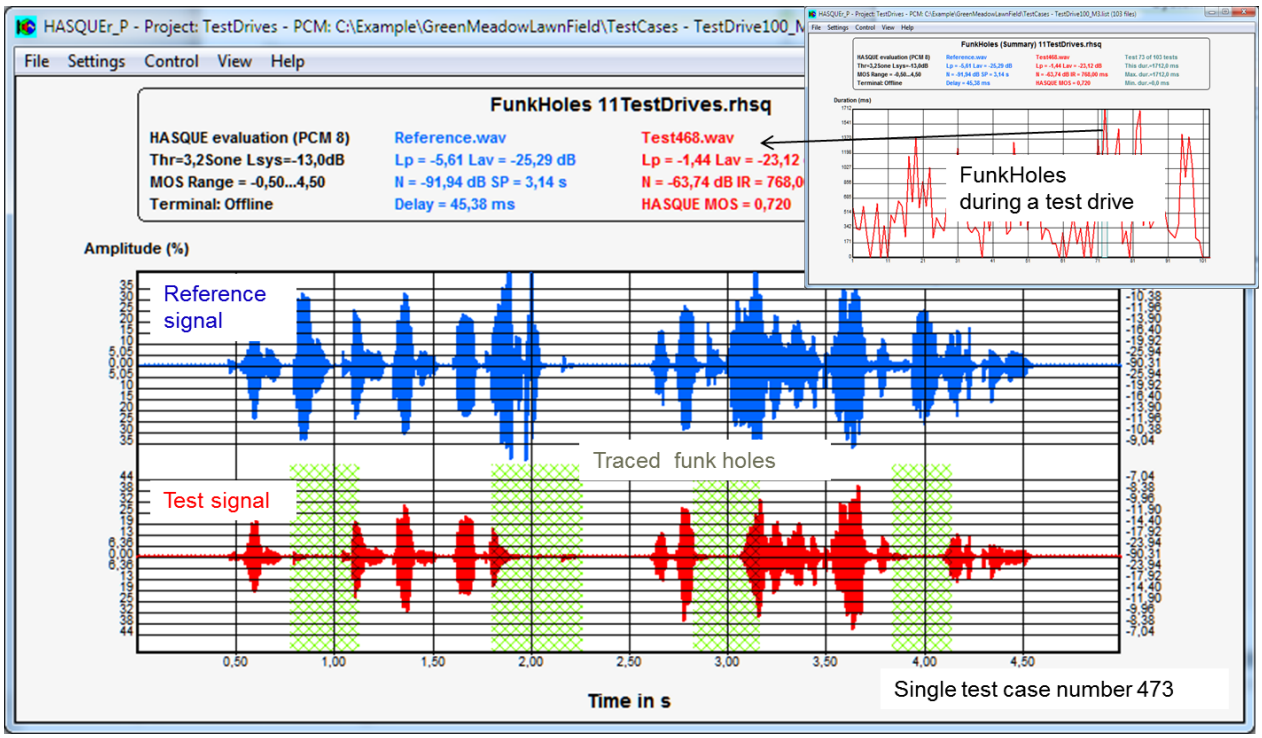

Bild 5: Signalunterbrechungen durch Funklöcher

Im kleinen Bild oben rechts in Bild 5 sind alle Fälle mit Störungen durch Signalauslöschungen aufgezeigt, die durch Funklöcher hervorgerufen werden können. Die Amplitude der roten Kurve zeigt die Zeitdauer der gesuchten Störung in jeder Aufnahme. Das übrige Bild zeigt die Darstellung eines ausgewählten

Testfalles aus dieser Übersichtsgraphik mit Bereichen, in denen Auslöschungen durch Funklöcher als „Traced func holes“ angezeigt werden.

The following property settings are indicated for the detection of funk holes.:

Signal Eigenschaften	Wert	Einheit
HControl.ThreshOfDistSone	0	Sone
HControl.ArtIntervall	100	ms
HControl.AddIRTimes	true	Bool
HControl.ThreshOfAttSone	-45	Sone
HControl.MaxCorrelation	86	%
HControl.ArtMOSThres	2.58	MOS
HControl.ArtSpecProperty	0	Factor
HControl.ArtSpecF1	0	Hz
HControl.ArtSpecF2	0	Hz

Einstellungen 6: Eigenschaften von Signalstörungen durch Funklöcher

Die Einstellungen müssen vor der Initialisierung und Evaluierung vorgenommen werden (Steuerung der Software).

Ergebnisse

Ergebnisse werden in der als HASQUEResults definierten Variable HResults zusammengefasst. Vektoren werden als Pointer übergeben. Bezugsgrößen eines Vektors (Länge, Zeit, Frequenz) werden mit zusätzlichen Variablen angegeben und sind über die Variablennamen einfach zuzuordnen.

Außer dem MOS-Wert werden bei der Qualitätsberechnung folgende zusätzliche qualitätsbestimmende Eigenschaften berechnet.

Bezeichnung	Format	Bedeutung
MOS	float	Qualitätsmaß gem. Hörtestparametrierung
SpeechDist	float	Sprachstörungen in dB(SPL)
PauseDist	float	Pausestörungen in dB(SPL)
NVarDist	float	Geräuschvarianz in dB(SPL) (Rauigkeit)
SPE_Error	*float	Vektor wahrnehmbarer Fehler in Sone
SPE_ErrorLen	int	Länge des Vektors
SPE_Delay	int	Verzögerung des Fehlersignals zu Referenzsignal
SamplesPerSPEFrame	int	Anzahl Abtastwerte pro SPE Wert
SPEFrameTime	float	Zeitintervall zwischen zwei SPE_error Abtastwerten (ms)

Ergebnisse 1: Qualitätsmaße und Fehler in HResults

Die Ergebnisse können im Anwenderprogramm als zusätzliche Information ausgegeben und weiterverwertet werden.

Anzeige des „hörbaren“ Fehlers

Der SPE_ERROR lässt sich im kartesischen System mit Y = Ordinate und X=Abszisse=Zeitachse in ms wie folgt abbilden: Y=SPE_Error[i] und X=i* SPEFrameTime für i=0, i< SPE_ErrorLen

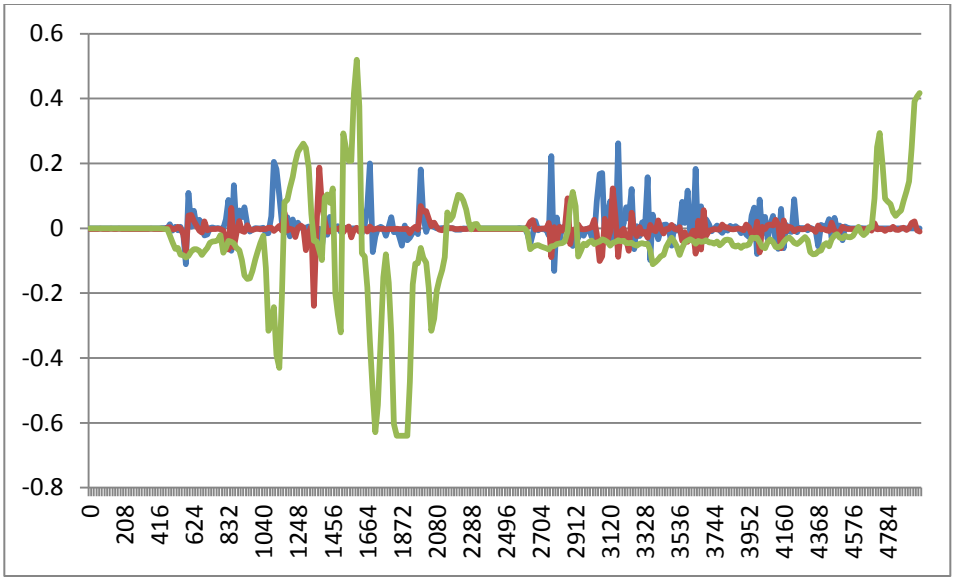

Bild 6: Darstellung des SPE_ERROR Vektors (grün) mit Referenz- und Testsignal

Zu beachten ist, dass sich die Zeitachse auf das Testsignal bezieht und gegenüber dem Referenzsignal um SPE_Delay verschoben ist. Die Abtastung des SPE_Error erfolgt wegen der notwendigen Zeitauflösung mit je einem Abtastwert pro SamplesPerSPEFrame Abtastwerte der tatsächlichen Abtastrate.

Signaleigenschaften

Mit Hilfe von zusätzlichen Messverfahre werden folgende Signaleigenschaften für das Referenz- und Testsignal ermittelt:

Bezeichnung	Format	Bedeutung
SAMLevelR	float	Spitzenpegel in dBov - Referenzsignal
StdLevelR	float	RMS Pegel in dBov - Referenzsignal
NoiseFloorR	float	Minimalpegel in dBov - Referenzsignal
RefSpectrum	*float	Betragsspektrum des Referenzsignals
RefSpectrumLen	int	Anzahl der Frequenzlinien
SAMLevelT	float	Spitzenpegel in dBov - Testsignal
StdLevelT	float	RMS Pegel in dBov - Testsignal
NoiseFloorT	float	Minimalpegel in dBov - Testsignal
TestSpectrum	*float	Betragsspektrum des Testsignals
TestSpectrumLen	int	Anzahl der Frequenzlinien
Frequency	*float	Frequenzvektor der Betragsspektren
TAL_Sample	int	Verzögerung des Testsignals in Abtastwerten
TAL_Time	float	Verzögerung des Testsignals in Sekunden

Ergebnisse 2: Signaleigenschaften in HResults der HASQUEDLLApi

Signaleigenschaften stehen in der HASQUEDLLApi nach der Evaluierung in HResults zur Verfügung.

Signalunterbrechung

Für Signalunterbrechungen stehen die Ergebnisse von Sprachunterbrechungen und Artefakten zur Verfügung.

Bezeichnung	Format	Bedeutung
SpeechInterrupts	long	Anzahl der Samples mit Sprachunterbrechung
SpeechActivity	long	Anzahl der Samples mit Sprachaktivität
SpeechInterruptsT	float	Zeit der Sprachunterbrechungen des Testsignals
SpeechActivityT	float	Zeit der Sprachaktivität des Referenzsignals
ClassErr	*float	Vektor der Artefaktindikation
isClassErr	bool	Zeigt mit true an, ob innerhalb einer Aufnahme Artefakte vorkommen

Ergebnisse 3: Signalunterbrechungen und klassifizierter Fehler

Die Sprachaktivität des Referenzsignals und die Sprachunterbrechungen des Testsignals innerhalb einer Aufnahme werden sowohl in Anzahl der Samples, als auch als Zeitwert in ms angegeben. Eine

prozentuale Angabe über Sprachunterbrechungen innerhalb einer Aufnahme lässt sich mit diesen Angaben einfach nachvollziehen.

Treten innerhalb einer Aufnahme Artefakte auf, wird dies mit der boolschen Variable isClassErr angezeigt.

Einen zeitlichen Bezug zum Beginn, die Dauer und dem Ende von Störungen durch Artefakte lässt sich mit Hilfe des Vektors *ClassErr* im kartesischen System mit Y = Ordinate und X=Abszisse=Zeitachse wie folgt abbilden.

Y= ClassErr [i] und X=i* SPEFrameTime für i=0, i< SPE_ErrorLen

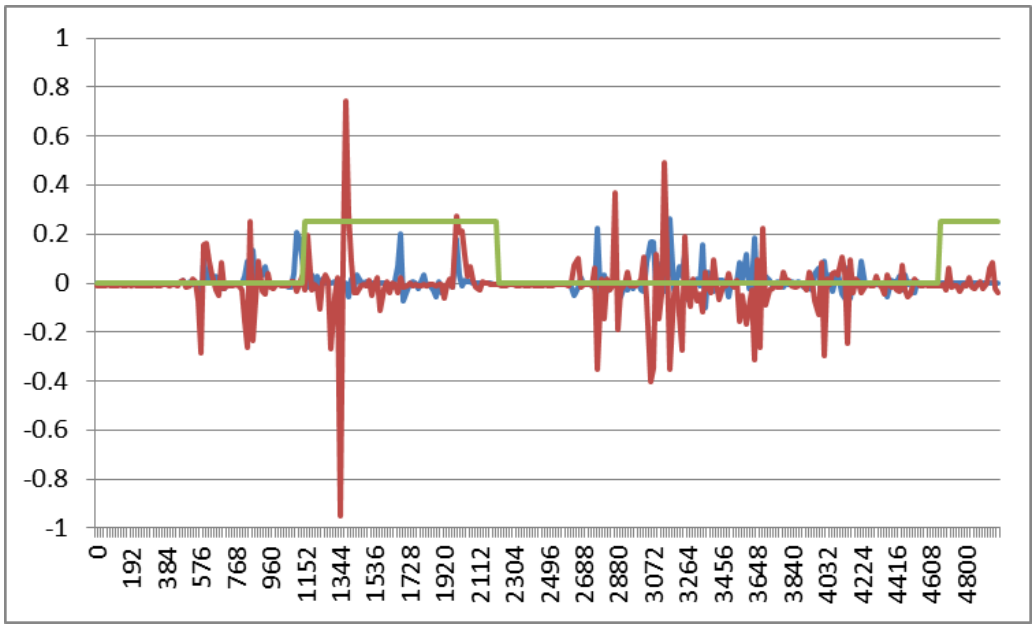

Bild 7: Darstellung des Klassifizierten Fehlers (grün) mit Referenz- und Testsignal

Zu beachten ist, dass sich die Zeitachse auf das Testsignal bezieht und gegenüber dem Referenzsignal um SPE_Delay verschoben ist. Die Abtastung der Erkennung von Artefakte erfolgt wegen der notwendigen Zeitauflösung mit je einem Abtastwert pro SamplesPerSPEFrame Abtastwerte der tatsächlichen Abtastrate.

Implementierung

Die Implementierung der DLL erfolgt typischerweise in einem vorhandenen Projekt, welches die zu bewertenden Referenz- und Testdaten bereitstellt, wie folgt.

1. Kopieren der Importbibliothek HASQUEDll.lib in den vorhandenen Projektordner und einbinden unter Projektmappe Eigenschaften→Linker/Eingabe/Zusätzliche Abhängigkeiten.
2. Kopieren der Dynamische Bibliothek HASQUEDll.dll in das Ausgabeverzeichnis der Anwendung.exe.
3. Kopieren der Quellen (HASQUEDLL.h, HASQUEDllApi.h, HASQUEDllApi.cpp) in das Quellenverzeichnis des Projekts und Einbinden der Quellen durch Projekt→Vorhandenes Element hinzufügen.
4. Objekt im Quellcode der Anwendung anlegen und anwenden (siehe Beispiel 1).

Einfache Anwendung mit nur zwei Programmzeilen

Mit dem Anlegen eines Objekts der Klasse HASQUEDllApi werden notwendige Speicher reserviert und die Defaultwerte für eine Qualitätsbeurteilung mit der Bewertungsskala für 8 kHz Samples nach ITU-T P.862 und den Hörtestparametern nach Anforderungen der BDBOS geladen. Die Klasse verwendet die verfügbaren Funktionen der DLL.
Mit Hilfe der HASQUEDllApi ist eine Qualitätsbeurteilung von 8 kHz Sprachdaten mit nur zwei Programmzeilen möglich (Beispiel 1).

```
#include "HASQUEDllApi.h"
HASQUEDllApi   hasqued;

hasqued.Init();              // Initialisierung
hasqued.RunHASQUE(xRef, xRefLen, xTest, xTestLen, xSR); // Evaluierung

//Die Ergebnisse stehen in hasqued.HResults zur Verfügung.
```

Beispiel 1: Objekt anlegen und anwenden

Im Beispiel 1 sind die blauen Übergabeargumente die vom Programmierer bereitzustellenden Variablen. xRef und xTest sind (float) Pointer auf die Referenz- bzw. Testsamples mit einer Länge von xRefLen bzw. xTestLen (long) Abtastwerten. xSR ist die gewünschte Abtastrate und kann für Standardanwendungen fest auf 8kHz gesetzt werden.

Erweiterte Anwendung

Steuerung der Software

Mit Hilfe der Steuerung der Software sind Anpassungen an von „Standardmessungen“ abweichenden Applikationen möglich. Wie in Beispiel 2 gezeigt, werden Steuervariable über HControl vor der Initialisierung gesetzt.

```
//Steuerung der Software: Beispiel Erweiterung des Latenzbereichs
hasqued.HControl.TALMax = 2;
hasqued.HControl.TALMin = -1;

//Evaluierung
hasqued.Init();             // Initialisierung
hasqued.RunHASQUE(xRef, xRefLen, xTest, xTestLen, xSR); // Evaluierung
```

Beispiel 2: Steuerung der Software erfolgt vor der Initialisierung

Parametrierung der Hörtestsimulation

Die Parametrierung der Hörtestsimulation erfolgt automatisch mit dem Anlegen des Objekts für die Qualitätsbeurteilung mit der Skalierung nach ITU-T P.862 und kann für die Anpassung an neue

```
hasqued.Init();             // Initialisierung

hasqued.HTestParams.BandwidthL = 300;     //lower cutoff frequency
hasqued.HTestParams.ThresholdOfAcceptance = (float)4.3;
hasqued.HTestParams.SystemLevel = 3;
hasqued.HTestParams.MOSmax = 5;           //Scaling max, MOS
hasqued.HTestParams.MOSmin = 1;           //Scaling min MOS
hasqued.HTestParams.Compressed = false;   //natural loudness dependent

hasqued.RunHASQUE(xRef, xRefLen, xTest, xTestLen, xSR); //Startet Evaluierung
```

Beispiel 3: Hörtestparameter werden nach der Initialisierung gesetzt

Eine Parametrierung der Hörtestbedingungen ist daher möglich, aber nicht grundsätzlich notwendig.

Darstellungen von Vektoren

Die kartesische Darstellung von Vektoren erfolgt zweidimensional mit der Abbildungsfunktion über der Zeit- oder Frequenzachse. Im Folgenden werden Programmbeispiele für die in der DLL verfügbaren Vektoren aufgezeigt. Blau markierte Variable sind vom Anwender bereitzustellende Punktearrays mit Y als Ordinate und X als Abszisse.

Darstellung der Frequenzgänge

```
for (int i = 0; i<hasqued.HResults.RefSpectrumLen; i++)
        {
        X[i] = hasqued.HResults.Frequency[i];
        YRef[i] = hasqued.HResults.RefSpectrum[i];
        YTest[i] = hasqued.HResults.TestSpectrum[i]);
        }
```

Beispiel 4: Erzeugung der Bildpunkte zur Darstellung der Referenz- und Testsignal- Spektren

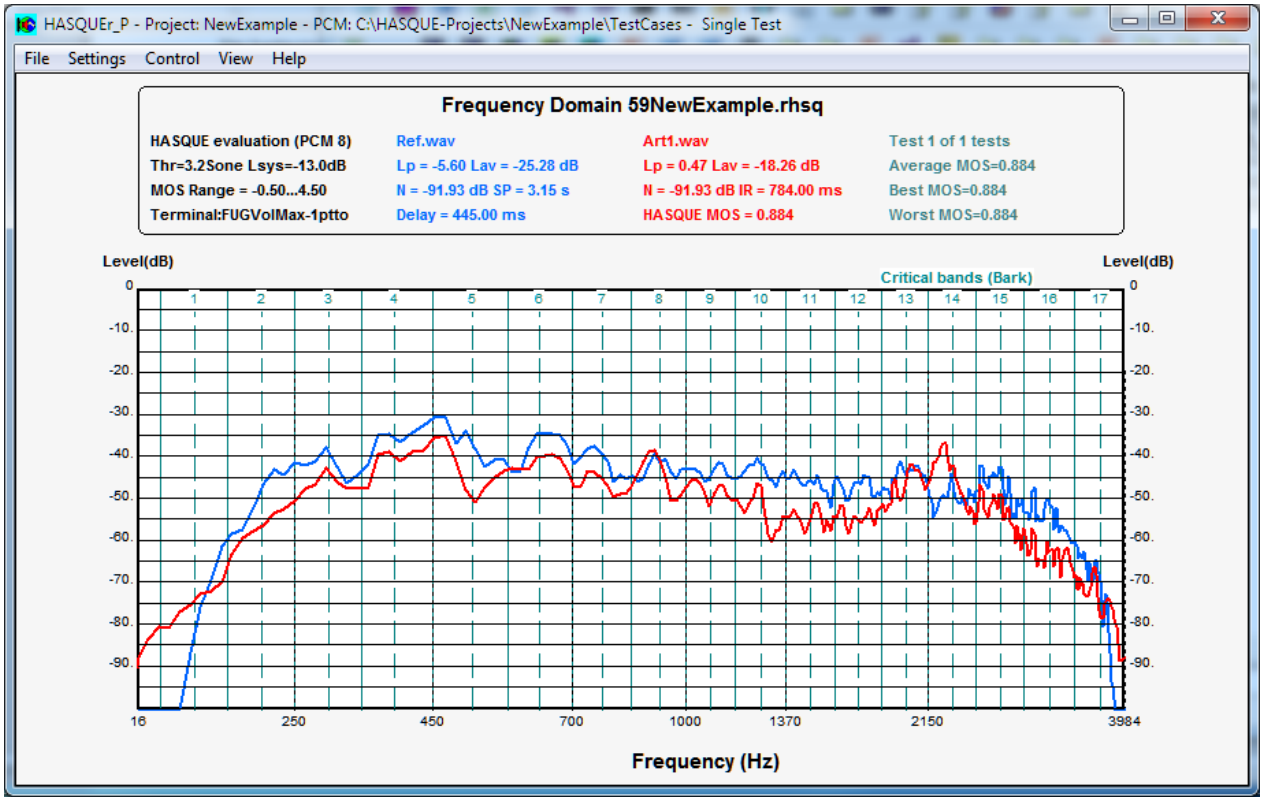

Bild 8: Spektrale Darstellung der Referenz- und Testsignale

Darstellung der hörbaren Fehler

Die Darstellung der hörbaren Fehler und den dazugehörenden Referenz- und Testsignalen über der Zeitachse verdeutlicht die zeitlichen Zusammenhänge zwischen den PCM-Abtastwerten und den Messwerten der hörbaren Fehler wie in Bild 6 gezeigt. Die Darstellung lässt sich nach Beispiel 5 erzeugen.

```
long k=0;
long l=0;
int j=0;
float SPENorm = 1/(float)64;
if(hasqued.HResults.TAL>0)
{
       l = hasqued.HResults.TAL;
}
else
{
       k = -hasqued.HResults.TAL;
}

for (int i = hasqued.HResults.SPE_Delay; i<hasqued.HResults. SPE_ErrorLen; i++)
       {
       X[i] = hasqued.HResults. SPEFrameTime *j; j++;
       YSPE[i] = hasqued.HResults. SPE_Error[i]*SPENorm;
       YRef[i] = xRef[k]);
       YTest[i] = xTest [l]* hasqued.HResults.GAL);
       k += hasqued.HResults.SamplesPerSPEFrame;
       l += hasqued.HResults.SamplesPerSPEFrame;
       }
```

Beispiel 5: Darstellung hörbare Fehler mit Referenz- und Testsignale

Der Zeitausgleich zwischen Referenz und Testsignal erfolgt mit Hilfe der zeitversetzten Indexzähler l und k, welche in Abhängigkeit der gemessenen Laufzeitverzögerung TAL gesetzt werden. Da das SPE_Error Signal wegen der erforderlichen Zeitauflösung zur Erfassung der hörbaren Unterschiede unterabgetastet wird, werden die Indexzähler der PCM-Signale(k,l) mit jeder Erhöhung des SPE_Error- Indexzählers (i) um SamplesPerSPEFrame erhöht. Da sich das Fehlersignal zeitlich mit dem Testsignal deckt, muss der Indexzähler i mit dem unterabgetasteten TAL-Wert SPE_Delay initialisiert werden, um eine Deckung mit dem Referenzsignal zu erreichen.

Der Amplitudenausgleich zwischen Referenz- und Testsignal erfolgt mit Hilfe des GAL - Wertes. Da die PCM - Signale auf 1 normiert sind, muss auch das Fehlersignal, welches in Sone angegeben wird mit Hilfe des Normierungsfaktors SPENorm auf 1 normiert werden.

Darstellung der Artefakte

Die zeitlichen Zusammenhänge zwischen den Abtastwerten der Artefakteerkennung und den Abtastwerten der Testsignale werden mit Beispiel 6 verdeutlicht. Das Programmbeispiel erzeugt mit X die Abtastzeitpunkte in ms welche zu den Abtastwerten der Artefakte YArt, der Referenz-(YRef) und Testignale (YTest) gehören (Bild 7).

```
long k=0;
long l=0;
int j=0;
if(hasqued.HResults.TAL>0)
{
       l = hasqued.HResults.TAL;
}
else
{
       k = -hasqued.HResults.TAL;
}

for (int i = hasqued.HResults.SPE_Delay; i<hasqued.HResults. SPE_ErrorLen; i++)
       {
       X[i] = hasqued.HResults. SPEFrameTime *j; j++;
       YArt[i] = hasqued.HResults. Artefacts [i];
       YRef[i] = xRef[k]);
       YTest[i] = xTest [l]* hasqued.HResults.GAL);
       k += hasqued.HResults.SamplesPerSPEFrame;
       l += hasqued.HResults.SamplesPerSPEFrame;
       }
```

Beispiel 6: Abtastzeitpunkte der Artefakte mit Referenz- und Testsignale

Der Zeitausgleich zwischen Referenz und Testsignal erfolgt mit Hilfe der zeitversetzten Indexzähler l und k, welche in Abhängigkeit der gemessenen Laufzeitverzögerung TAL gesetzt werden. Da die Erkennung von Artefakten mit einer Zeitauflösung von SPEFrameTime arbeitet, werden die Indexzähler der PCM-Signale(k,l) mit jeder Erhöhung des Indexzählers (i), mit dem die Zeitpunkte der Artefakteerkennung abgetastet werden, um SamplesPerSPEFrame erhöht. Da sich das Fehlersignal zeitlich mit dem Testsignal deckt, muss der Indexzähler i mit dem unterabgetasteten TAL-Wert SPE_Delay initialisiert werden, um eine Deckung mit dem Referenzsignal zu erreichen.

Der Amplitudenausgleich zwischen Referenz- und Testsignal erfolgt mit Hilfe des GAL - Wertes. Die Abtastwerte des ClassErr - Vektors betragen 0.25 in den Zeitintervallen, in denen Artefakte erkannt wurden und sind sonst auf 0 gesetzt.

Spezifikationen

Allgemeines

DLL 32 Bit für Windows Betriebssysteme
Abtastrate: frei wählbar – Default 8kHz
Bewertungsskala: programmierbar – Default für 8 kHz Samples nach ITU-T P.862
Hörtestparameter: programmierbar - Default nach Anforderungen der BDBOS

Signalverzögerung (Latenz)

Maximale Verzögerung 1000 ms (programmierbar)
Minimale Verzögerung -200 ms (programmierbar)
Zeitvarianz innerhalb eines Testsamples: $\underline{<}$ 50 ms

Sprachsamples (Referenzsignal)

In Anlehnung an ITUT-P.862 gelten folgende Richtwerte:

Sprachpegel: L(peak) typ -6 dBov, L(avarage) typ. -30 dBov
Beginn der ersten Sprachäußerung >500 ms (>Latenz) nach Beginn der Aufnahme
Ende der letzten Sprachäußerung >500 ms (>Latenz) vor dem Ende der Aufnahme
Sprachaktivität >40 - <80 % der Aufnahme
Aufnahmelänge: 5-10 Sekunden
Signal -Störabstand: > 50 dB
Noise floor: >90dBov bei 32 Bit >75dBov bei 16 Bit - Abtastwerte sollen keine 0 – Folgen aufweisen.

Übergabeargumente Signalbuffer

Referenz- und Testsignal	: Pointer 32 Bit float
	: Signalaussteuerung $\underline{+}$ (1-x) = Vollaussteuerung (0dBov)
	: 1- x=1- $1/(2^{31})$=0.9999…;
Anzahl der Referenzsamples	: long
Anzahl der Testsamples	: long
Abtastrate	: int

Steuerung von Funktionen

Die DLL wird automatisch mit Standardwerten initialisiert.
Die Steuerung kann bei Bedarf individuell mit Hilfe der für die Steuerung relevante und als HASUEControl definierte Variable zur Steuerung von Funktionen und zur Festlegung von Schwellenwerten und Wertebereichen an individuelle Anforderungen angepasst werden.

```
typedef struct
{
//Gain compensation by gain alignment (GAL)
	bool	isGAL;		//adaptive GAL if true, else fixed GAL
	float	GainCorrDeg;	//gain factor for fixed GAL
//Latency compensation by time alignment (TAL)
	bool	isTAL;		//adaptive TAL if true, else fixed TAL
	int		Delay_p2;	//Number of samples for fixed TAL
	float	TALMax;		//maximum TAL in seconds
	float	TALMin;		//minimum TAL in seconds
	bool	isBlockDComp;	//extended TAL with block compensation
	bool	isJitterDComp; //optional extended TAL with latency jitter
//Signal interrupts
	float	SPIRLOUDTHR1; //Threshold in Sone interpreted as SPIR
	float	SPIRMinTime1; //Minimum time in ms interpreted as SPIR
	float	SPIRLOUDTHR2; //Threshold 2 not applied yet for DLL
	float	SPIRMinTime2;	//Minimum time 2 not applied yet for DLL
//Properties of the dignal distortions for a certain error clasification

	float	ThreshOfDistSone;		//Threshold of additionally disturbance
(Sone)
	float	ArtIntervall; //Interval time
	bool	AddIRTimes;		//include times with interrupts belonging to
artefacts
	float	ThreshOfAttSone; //Threshold of attenuations (Sone)
	float	MaxCorrelation;		//maximum expected correlation with the original
	float	ArtMOSThres;	//MOS Threshold for proper artifical signal extensions
	float	ArtSpecProperty; //spectral property AKF - higeh value == narrowband
	int		ArtSpecF1;		//first base frequency in Hz
	int		ArtSpecF2;		//second base frequqncy in Hz

//Others
	bool	SkipLicenseWarning;	//skips warning message before license time has
	//finished if true
}HASQUEControl; //Control HASQUE functions
```

Struktur 1: Steuervariablen

Parametrierung der Hörtestsimulation

Die DLL wird automatisch mit der Bewertungsskala für 8 kHz Samples nach ITU-T P.862 und den Hörtestparametern nach Anforderungen der BDBOS initialisiert.

```
typedef struct
{
	int SR;
	float ThresholdOfAcceptance; //threshold of max. accepted audible error in sone
	float UpperFC;		//upper cutoff frequency - not applied
	float LowerFC;		//lower cutoff frequency
	float SystemLevel; //listening loudness level 0dB = nominal ...speech = -13dB...
	float MOSmax; //maximum MOS - excellent
	float MOSmin; //minimum MOS - bad
	bool Compressed;	//error weighting true = acc. to ITU-T P.862 - false =
				natural loudness
}ListeningTestParams; //Listening test conditions and MOS scaling
```

Struktur 2: ListeningTestParams

Ergebnisse

Der Zugriff auf Ergebnisse erfolgt mit Hilfe der angelegten Ergebnisstruktur.

```
typedef struct
{
	float  SAMLevelR;    //peak level of the reference signal in dBov
	float  StdLevelR;    //RMS level of the reference signal in dBov
	float  NoiseFloorR;  //minimum level of the reference signal in dBov
	float  *RefSpectrum;        //discrete spectral magnitudes of the reference spectrum
	int    RefSpectrumLen;      //number of frequency bins of the reference spectrum

	float  SAMLevelT;                  //peak level of the test signal in dBov
	float  StdLevelT;                  //RMS level of the test signal in dBov
	float  NoiseFloorT;         //minimum level of the test signal in dBov
	float  *TestSpectrum;              //discrete spectral magnitudes of the test spectrum
	int    TestSpectrumLen;     //number of frequency bins of the test spectrum

	float  *Frequency;   //discrete frequencies[0...N] belonging to the spectra[0...N]
	float  GAL;   //gain alignment factor
	int    TAL_Samples;  //time alignment in samples
	float  TAL_Time;     //time alignment in seconds

	float  Mos;   //estimated mean opinion score
	float  SpeechDist;   //total distortions during speech activity in dB(SPL)
	float  PauseDist;    //total distortions during speech pause in dB(SPL)
	float  NVarDist;     //total noise variant distortions in dB(SPL)
	float  *SPE_Error;   //subsampled signed audible errors (Sone)
	int    SPE_ErrorLen; //total number of signed audible error samples
	int    SPE_Delay;    //delay of the error samples related to the reference signal
	int    SamplesPerSPEFrame;                //number of samples per audible error sample
	float  SPEFrameTime;               //time per audible error sample in ms

	long   SampleRate;                 //samples per second
	long   SpeechInterrupts;    //number of samples interrupted
	long   SpeechActivity;             //Number of Samples within speech activity
	float  SpeechInterruptsT;   //Speech interrupt time in seconds
	float  SpeechActivityT;            //Speech activity time in seconds

	float  *ClassErr;    // classified error with SPE_ErrorLen and SPEFrameTime
	long   isClassErr;   //true if artefact detected, else false

	char   ReferenceNumber[MAX_PATH];  //registration number
	int    LicenseNofDays;      //remaining license in days
	char   EndOfLicense[MAX_PATH];     //The license is valid until this date
} HASQUEResults; //Results
```

Struktur 3: HASQUEResults

Literatur

1. E. Zwicker: „Psychoakustik“ Springerverlag 1982, ISBN 3-540-11401-7.

2. E. Zwicker, H. Fastl : „Psycho acoustics“, Springerverlag 1999, ISBN 3-540-65063-6 .

3. ITU-T P.862 Recommendation: “Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrow band telephone networks and speech codecs”, 02/2001

4. ITU-T P.862.1 Recommendation: “Mapping function for transforming P.862 raw result scores to MOS-LQO“, 11/2003

5. ITU-T P.862.3 Recommendation: “Application guide for objective quality measurement based on Recommendations P.862, P.862.1 and P.862.2”, 11/2007

6. E. Terhardt: “Fourier Transformation of Time Signals”, Acustica, Vol. 57, 1985.

7. Rolf Kapust: „Qualitätsbeurteilung codierter Audiosignale mittels einer BARK – Transformation. Dissertation“, Technische Fakultät der Universität Erlangen, 1993.

8. Michael Walker: „Gehöradäquate digitale Sprachsignalverarbeitung“, Funkschau 04/2003, ISSN 0016-2841. Seite 57-58.

9. Michael Walker: HASQUE “Vorrichtung und Verfahren für eine gehöradäquate objektive Qualitätsschätzung von Audiosignalen” DE 10 2005 019 903 A1 2006.11.02, 29.4.2005

Anhang:

Abbildungen

Einstellungen

Beispiele

Ergebnisse

Datenstrukturen