SEGMENTACIÓ AUTOMÀTICA DE LA RESPIRACIÓ A PARTIR DE L'ENREGISTRAMENT ACÚSTIC

per

Juan Castro Mayorgas

Memòria tècnica corresponent al
Treball Final de Màster de:

MÀSTER INTERUNIVERSITARI D'ENGINYERIA DE TELECOMUNICACIÓ

Universitat Oberta de Catalunya / Universitat Ramon Llull

Consultor: Pere Martí Puig

Gener 2015

© Juan Castro Mayorgas, 2015
Tots els drets reservats

Els textos, imatges i codi publicats en aquesta obra estan subjectes –llevat que s'indiqui el contrari– a una llicència de Reconeixement-CompartirIgual 4.0 Internacional (BY-SA) de Creative Commons. Està permès la còpia, adaptació, distribució i transmissió en qualsevol medi o format, per qualsevol propòsit, inclús comercial, sempre que es citi l'autor i es distribueixi sota la mateixa llicència que l'original. La llicència completa es pot consultar a http://creativecommons.org/licenses/by-sa/4.0/legalcode.

Als meus pares

RESUM

SEGMENTACIÓ AUTOMÀTICA DE LA RESPIRACIÓ A PARTIR DE L'ENREGISTRAMENT ACÚSTIC

per

Juan Castro Mayorgas

MÀSTER INTERUNIVERSITARI D'ENGINYERIA DE TELECOMUNICACIÓ
Universitat Oberta de Catalunya / Universitat Ramon Llull

Aquest Treball Fi de Màster desenvolupa un programari per a identificar, en temps real, els moviments respiratoris -inspiració i espiració- a través d'un micròfon. El programari, desenvolupat en Matlab i anomenat ASBSLAB per a la versió gràfica i ASBSLABCONSOLE per a la versió a la línia d'ordres, és fruit d'un procés de recerca i experimentació.

S'han enregistrat 48 minuts de moviments respiratoris, que corresponen a la respiració de quatre voluntaris, per analitzar-los, extreure'ls-hi 18 característiques acústiques i generar un model de dades.

S'ha dissenyat un primer nivell d'identificació basat en la classificació de mostres a partir del mètode supervisat kNN. Les proves de validació creuada han donat uns resultats de més del 95 % d'encerts i la validació amb un conjunt de mostres alienes al repositori ha donat un percentatge d'encerts prop del 70 %. A més a més, s'ha fet una validació amb totes les combinacions possibles de característiques acústiques i s'ha generat una taula de combinacions candidates a donar els millors resultats.

S'ha implementat una màquina d'estats basada en la seqüència temporal de resultats del mètode de classificació. La màquina d'estats pren la decisió final del tipus de moviment

respiratori escoltat. Les proves realitzades donen un percentatge d'identificacions positives de més del 93 %.

ASBSLAB permet configurar el conjunt de característiques acústiques que es vol fer servir en una sessió d'identificació. A més, està preparat per a utilitzar les freqüències de mostreig 22.050 Hz, 16.000 Hz, 11.025 Hz i 8.000 Hz, amb longituds de finestra de 128 ms, 96 ms, 92 ms, 69 ms, 64 ms, 46 ms i 23 ms. Si bé els millor resultats s'aconsegueixen amb les opcions 22.050 Hz – 92 ms i 16.000 Hz – 128 ms, es deixen totes les opcions a l'abast de l'usuari per a què pugui experimentar amb tots els paràmetres.

Paraules clau: Processament digital del senyal, reconeixement per patrons, anàlisi de la respiració en temps real, identificació de la inspiració i espiració, característiques acústiques.

ABSTRACT

AUTOMATIC SEGMENTATION OF BREATHING
THROUGH ACOUSTIC RECORDING

by

Juan Castro Mayorgas

TELECOMMUNICATION ENGINEERING UNIVERSITY MASTER'S DEGREE
Universitat Oberta de Catalunya / Universitat Ramon Llull

This work develops an application to identify, in real time, the respiratory movements - inspiration and expiration- through a microphone. The application, developed in Matlab and called ASBSLAB for the GUI version and ASBSLABCONSOLE for the command-line version, is the result of a research and experimentation process.

A total of 48 minutes of breathing movements were recorded, which correspond to the breathing of four volunteers. The recording were analyzed, eighteen acoustic features were extracted and a data model were generated.

A first level of identification were designed, based on the samples classification through kNN method. Cross-validation testing yielded results of more than 95 % of positive classifications. Further validation with a set of samples outside the repository gave almost 70 % of positive identifications. In addition, a validation with every possible combination of acoustic characteristics were executed and the results were included in a table. The table suggests likely combinations to deliver best results.

A state-machine based on the temporal sequence of previous results were implemented. The state-machine takes the final decision of the type of respiratory movement heard. The tests give a percentage of positive identifications above 93 %.

ASBSLAB lets the user configure the set of acoustic characteristics that wants to use in a running session of the application. In addition, the application is ready to work with the sampling frequencies of 22,050 Hz, 16,000 Hz, 11,025 Hz, 8000 Hz and window sizes of 128 ms, 96 ms, 92 ms, 69 ms, 64 ms, 46 ms and 23 ms. Although the best results are achieved with the options 22,050 Hz - 92 ms and 16,000 Hz - 128ms, all options are available to let the user experiment with every possible combinations.

Keywords: Digital signal processing, pattern recognition, respiratory analysis in real time, identification of inspiration and expiration, acoustic features.

RESUMEN

SEGMENTACIÓN AUTOMÁTICA DE LA RESPIRACIÓN A PARTIR DE LA GRABACIÓN ACÚSTICA

por

Juan Castro Mayorgas

MÁSTER INTERUNIVERSITARIO DE INGENIERÍA DE TELECOMUNICACIÓN
Universitat Oberta de Catalunya / Universitat Ramon Llull

Este Trabajo Fin de Máster desarrolla una aplicación para identificar, en tiempo real, los movimientos respiratorios -inspiración y espiración- a través de un micrófono. La aplicación, desarrollada en Matlab y denominada ASBSLAB para la versión gráfica y ASBSLABCONSOLE para la versión de la línea de comandos, es fruto de un proceso de investigación y experimentación.

Se ha grabado un total de 48 minutos de movimientos respiratorios, que corresponden a la respiración de cuatro voluntarios, para analizarlos, extraer sus características acústicas y generar un modelo de datos.

Se ha diseñado un primer nivel de identificación basado en la clasificación de muestras a partir del método kNN. Las pruebas de validación cruzada han dado unos resultados de más del 95 % de aciertos y la validación con un conjunto de muestras ajeno al repositorio ha dado un porcentaje de aciertos de casi el 70 %. Además, se ha ejecutado una validación con todas las posibles combinaciones de características acústicas y se ha generado una tabla con las candidatas a ofrecer los mejores resultados.

Se ha implementado una máquina de estados basada en la secuencia temporal de resultados del método de clasificación. La máquina de estados toma la decisión final del tipo de

movimiento respiratorio escuchado. Las pruebas realizadas proporcionan un porcentaje de identificaciones positivas superiores al 93 %.

ASBSLAB permite configurar el conjunto de características acústicas que se desean utilizar en una sesión de identificación. Además, la aplicación está preparada para trabajar con las frecuencias de muestreo 22.050 Hz, 16.000 Hz, 11.025 Hz, 8.000 Hz y con anchos de ventana de 128 ms, 96 ms, 92 ms, 69 ms, 64 ms, 46 ms y 23 ms. Si bien los mejores resultados se consiguen con las opciones 22.050 Hz – 92 ms y 16.000 Hz – 128 ms, se dejan todas las opciones al alcance del usuario para que pueda experimentar con todos los parámetros de configuración.

Palabras clave: Procesamiento digital de la señal, reconocimiento por patrones, análisis de la respiración en tiempo real, identificación de la inspiración y espiración, características acústicas.

AGRAÏMENTS

Primer de tot, vull donar l'agraïment al meu consultor, Pere Martí Puig, pel seu suport i orientació durant el transcurs del Treball Fi de Màster. També vull agrair-li de forma especial que hàgim realitzats connexions periòdiques per videoconferència amb la finalitat de comentar l'estat en curs del Treball, amb tots els avantatges que representa aquest mitjà de comunicació en eficàcia, facilitat de diàleg i estalvi de temps davant el correu electrònic.

Com no, agraeixo a Mayte, la meva parella, la seva infinita paciència perquè ha sabut entendre que algunes etapes del Treball requerien a vegades un esforç addicional que m'aïllaven durant llargues hores davant l'ordinador.

També, haig d'agrair la col·laboració d'en Marc, Olga, Àngel, Enric, Miquel i Mayte (un altre cop). Ells m'han donat 'aire' per dur a terme el Treball, participant de forma activa com a voluntaris en l'enregistrament de la seva respiració per a la generació dels model i registres de proves.

I per acabar, dono gràcies als meus pares, germans i resta de la família que, d'una manera o d'altra, m'han donat suport durant aquest temps.

ÍNDEX DE CONTINGUTS

Pàg.

RESUM ..... **iv**

ABSTRACT ..... **vi**

RESUMEN ..... **viii**

AGRAÏMENTS ..... **x**

ÍNDEX DE CONTINGUTS ..... **xi**

ÍNDEX DE FIGURES ..... **xiii**

ÍNDEX DE TAULES ..... **xv**

CAPÍTOL 1 - INTRODUCCIÓ **1**

1.1. JUSTIFICACIÓ, PUNT DE PARTIDA I APORTACIÓ 1

1.2. OBJECTIUS 2

1.3. ENFOCAMENT I METODOLOGIA 3

1.4. PLANIFICACIÓ 3

1.5. ANÀLISI DE RISCOS 8

1.6. PRODUCTES OBTINGUTS 8

1.7. BREU DESCRIPCIÓ DELS ALTRES CAPÍTOLS 8

CAPÍTOL 2 - ANTECEDENTS **10**

CAPÍTOL 3 - ANÀLISI DEL SO RESPIRATORI **12**

3.1. ENREGISTRAMENT I FILTRATGE 12

3.1.1. Introducció 12

3.1.2. Escenari d'enregistrament 13

3.1.3. Procés d'enregistrament 14

3.1.4. Filtratge 20

3.2. EXTRACCIÓ DE CARACTERÍSTIQUES 23

3.2.1. FFT 23

3.2.2. Segmentació i encavalcament 23

3.2.3. Enfinestrat 25

3.2.4. Preparació 26

3.2.5. Característiques 27

3.3. GENERACIÓ DEL REPOSITORI 39

CAPÍTOL 4 - SISTEMA D'IDENTIFICACIÓ **43**

4.1. INTRODUCCIÓ 43
4.2. SELECCIÓ DEL SISTEMA DE CLASSIFICACIÓ 44
4.3. IMPLEMENTACIÓ DEL SISTEMA DE CLASSIFICACIÓ 45
4.3.1. estandardització 46
4.4. VALIDACIÓ DEL REPOSITORI 47
4.4.1. validació creuada 47
4.4.2. Combinacions de característiques 53
4.4.3. Validació amb mostres desconegudes 57
4.5. SEQÜÈNCIA TEMPORAL 60
4.6. VALIDACIÓ 63
4.7. COST COMPUTACIONAL 65
CAPÍTOL 5 - INTERFÍCIE D'ÚS 69
5.1. INTRODUCCIÓ 69
5.2. ASBSLABCONSOLE 69
5.2.1. Proves d'execució 70
5.3. ASBSLAB 73
CAPÍTOL 6 - CONCLUSIONS 76
6.1. VISIÓ GENERAL I CONCLUSIONS 76
6.2. LÍNIES FUTURES DE TREBALL 78
BIBLIOGRAFIA 80
ANNEX A – CODI FONT 82
VITA 134

ÍNDEX DE FIGURES

Figura 1. Diagrama Gantt del TFM 7
Figura 2. Exemple de senyal analògic 12
Figura 3. Exemple de senyal analògic i la seva corresponent digitalització 12
Figura 4. Escenari d'enregistrament 14
Figura 5. Diagrames temporal i espectral de respiració bucal (voluntari 1) 15
Figura 6. Diagrames temporal i espectral de respiració bucal (voluntari 2) 15
Figura 7. Diagrames temporal i espectral de respiració bucal (voluntari 3) 16
Figura 8. Diagrames temporal i espectral de respiració bucal (voluntari 4) 16
Figura 9. Diagrames temporal i espectral de respiració nasal (voluntari 1) 17
Figura 10. Diagrames temporal i espectral de respiració nasal (voluntari 2) 17
Figura 11. Diagrames temporal i espectral de respiració nasal (voluntari 3) 18
Figura 12. Diagrames temporal i espectral de respiració nasal (voluntari 4) 18
Figura 13. Exemple d'espectre de potència de soroll enregistrat 20
Figura 14. Resposta en freqüència del filtre el·líptic dissenyat 21
Figura 15. Diagrames temporal i espectral una vegada aplicat el filtre (voluntari 1) 22
Figura 16. Detall de l'efecte del filtre 22
Figura 17. Enfinestrat Hamming sobre un senyal mostra i efecte resultant 26
Figura 18. Energia de llindar 28
Figura 19. Histograma del *Spectrum Centroid* 30
Figura 20. Histograma del *Spectrum Flatness*. 31
Figura 21. Histograma del *Zero-Crossing Rate* 32
Figura 22. Histograma del *Spectrum RollOff* 33
Figura 23. Mitja d'acumulat de magnitud en funció de la freqüència 34
Figura 24. Histograma del ratio energia acumulada a 2.000 Hz 35
Figura 25. Relació Hz - Mel 36
Figura 26. Banc de filtres Mel 37
Figura 27. Histogrames dels MFCC 39
Figura 28. Procés d'aprenentatge (capítol 3) 43
Figura 29. Procés d'identificació 43
Figura 30. Encerts segons característiques i el seu nombre. 54
Figura 31. Fases de creixement i decaïment 61
Figura 32. Diagrama d'estats 62

Figura 33. Cost del mètode kNN en funció del nombre de característiques..... 66
Figura 34. Interfície de l'asbslab..... 73
Figura 35. Selecció del dispositiu d'entrada..... 74
Figura 36. Selecció de la configuració del model de dades..... 74
Figura 37. asbslab en execució..... 75

ÍNDEX DE TAULES

Taula 1. Taula de fites del TFM 6
Taula 2. Propostes de longitud de finestra en funció de la freqüència de mostreig 24
Taula 3. Relació de freqüències del banc de filtres 37
Taula 4. Noms dels repositoris en funció del mostreig i finestra 40
Taula 5. Identificació de les columnes del repositori 41
Taula 6. Codi numèric assignat als moviments respiratoris 42
Taula 7. Resultats de la validació creuada 48
Taula 8. Percentatges d'identificació per a la versió en 22.050 Hz del model 49
Taula 9. Percentatges d'identificació per a la versió en 16.000 Hz del model 50
Taula 10. Percentatges d'identificació per a la versió en 11.025 Hz del model 51
Taula 11. Percentatges d'identificació per a la versió en 8.000 Hz del model 51
Taula 12. Millors i pitjors combinacions de característiques 55
Taula 13. Millors i pitjors combinacions de característiques (sense RMS) 56
Taula 14. Resultats de validació amb mostres desconegudes 57
Taula 15. Matriu de confusió per a mostres desconegudes 58
Taula 16. Millors i pitjors combinacions de característiques amb mostres desconegudes 59
Taula 17. Identificacions positives amb característiques aïllades 63
Taula 18. Identificacions positives amb característiques combinades 64
Taula 19. Temps de computació amb 22.050 Hz i 92 ms de finestra 66
Taula 20. Temps de computació amb 22.050 Hz i 23 ms de finestra 67
Taula 21. Paràmetres d'execució d'asbslabconsole 69

CAPÍTOL 1 - INTRODUCCIÓ

El Treball Final de Màster (TFM) és, tal com indica el pla docent, "*l'escenari perquè l'estudiant posi en pràctica la integració de les competències, genèriques i específiques adquirides durant el Màster"* i afegeix que "*així mateix, es busca que l'estudiant pugui anar més enllà i abordar un treball, que sense ser necessàriament de recerca, pugui demostrar la seva capacitat per abordar coneixements addicionals i complementaris als quals ha adquirit en els seus estudis*".

Aquest Treball, ubicat dintre de l'àrea temàtica del processament de senyal conté un elevat component de recerca i, al mateix temps que aborda coneixements complementaris als propis del màster, desenvolupa un producte final fruit del procés de recerca.

1.1. JUSTIFICACIÓ, PUNT DE PARTIDA I APORTACIÓ

El processament del senyal està molt present en la medicina. Les seves aplicacions a qualsevol especialitat per a la diagnosi, monitorització o exploració són innumerables. Aplicat a l'anàlisi de la respiració, el processament digital del so proveeix de nous mecanismes d'estudi de les vies respiratòries i de detecció de possibles alteracions i malalties com l'apnea o l'asma entre altre moltes.

L'enfocament emprat en el processament digital del so respiratori ha estat basat segons un criteri d' 'enregistrament inicial – anàlisi posterior'. És a dir, l'enregistrament del so respiratori es fa en una primera fase i posteriorment s'analitza el conjunt enregistrat. Aquest model permet la utilització d'unes tècniques que en un escenari d'anàlisi en temps real no serien aplicables. L'objectiu d'aquest Treball, és omplir aquest buit, amb l'aportació d'una proposta d'anàlisi del so respiratori en temps real.

A partir d'un escenari i condicions inicials, aquest Treball analitza el so respiratori amb la finalitat de detectar les fases d'inspiració i espiració, ja sigui bucal o nassal. Per fer-ho, es fa l'enregistrament acústic de la respiració a través d'un micròfon situat a una certa distància física de l'usuari.

Aquest plantejament, fa possible pensar en sistemes de monitorització no invasiva de la respiració en diverses aplicacions pràctiques com, per exemple:

- Mesurar el nivell d'estrès.
- Monitoritzar la respiració d'un usuari amb asma per a controlar que està respirant únicament pel nas per prescripció del pneumòleg.
- Monitoritzar el patró de respiració d'un usuari en la pràctica de tècniques de relaxació i avisar-li quan no segueix al patró prèviament definit.

1.2. OBJECTIUS

L'objectiu principal d'aquest treball és implementar una solució per a detectar, en temps real, els moviments respiratoris d'inspiració i espiració, bucal o nasal i en qualsevol combinació, a través d'un micròfon situat a una certa distància de l'usuari.

Per l'assoliment d'aquest objectiu principal, és necessari assolir també els següents objectius parcials:

- Cercar literatura existent al voltant de l'anàlisi de la respiració, per conèixer plantejaments variats, els reptes afrontats i les diferents propostes a cada un dels problemes plantejats.
- Establir un escenari inicial sobre el qual es desenvolupa tota la recerca i la solució proposada.
- Conèixer, analitzar i seleccionar el conjunt d'atributs adients que millor caracteritzin el so respiratori per l'objectiu del Treball.
- Escollir i implementar el sistema de decisió que millor s'adapti a les necessitats del Treball.
- Generar un model de dades a partir de l'anàlisi i atributs de mostres dels moviments respiratoris creades de forma supervisada.
- Generar coneixement a partir de la memòria del Treball, on es reflecteixen els resultats experimentals en totes les fases dutes a terme.
- Generar el codi font de programació de la solució proposada i de les eines utilitzades per a l'anàlisi.

Per altra banda, és important indicar que no són objectius d'aquest Treball:

- La detecció simultània dels moviments respiratoris simultanis boca-nas.
- La detecció de patologies respiratòries.

1.3. ENFOCAMENT I METODOLOGIA

L'enfocament d'aquest Treball és molt pràctic, seguint una metodologia molt experimental que permet analitzar els reptes inherents en cada fase del Treball i treure les conclusions més adients d'acord a les necessitats de la solució. D'acord a això, la mateixa memòria està organitzada linealment segons el procés de recerca, experimentació i desenvolupament de la solució final.

Així mateix, s'ha considerat oportú, pel mateix ingredient de recerca que suposa el Treball, que la memòria tècnica s'escrigués en simultani al desenvolupament del Treball, registrant i documentant cada un dels passos en l'assoliment dels objectius del Treball.

1.4. PLANIFICACIÓ

El treball es desglossa en l'elaboració de les següents activitats, que, a excepció de la memòria, que es redacta en paral·lel amb cada una de les activitats, es desenvolupen de forma consecutiva:

Codi 1.1	Definició del TFM
Durada	11 dies
Descripció	Definició del TFM, consensuat entre les diverses opcions proposades. Inclou la cerca bibliogràfica i la definició de l'abast.

Codi 1.2	Elaboració del Pla de Treball
Durada	7 dies
Descripció	Definició dels objectius del TFM, el seu desglossament en tasques i la seva planificació temporal.

Codi 2.0	Definició del marc de treball
Durada	4 dies
Descripció	Definició i selecció dels elements físics i eines de programari necessaris per a l'elaboració del TFM.

Codi 3.1	Enregistrament i filtratge
Durada	5 dies
Descripció	Enregistrament del so respiratori de diferents persones amb varietat d'edat i gènere. Estudi dels procediments més adients a aplicar abans i després de l'enregistrament per a obtenir la millor relació senyal/soroll. L'objectiu és obtenir una mostra representativa de sons respiratoris identificats sobre els que poder fer l'anàlisi de característiques.

Codi 3.2	Estudi i selecció d'atributs
Durada	9 dies
Descripció	A partir de les gravacions representatives, s'han d'estudiar els atributs i paràmetres de segmentació que puguin ser rellevants en la identificació dels moviments respiratoris. Aquesta activitat inclou anàlisi tant en el domini freqüencial com en el temporal.

Codi 3.4	Generació de repositori
Durada	2 dies
Descripció	Generació del repositori de mostres representatives amb els seus atributs. La implementació ha de permetre la flexibilitat de generar repositori amb la variabilitat de la grandària de la finestra de temps de segmentació.

Codi 5.1	Selecció sistema de decisió
Durada	5 dies
Descripció	Selecció del sistema de decisió més adequat pel propòsit del Treball en funció de paràmetres tals com el nombre de mostres del repositori, el nombre de característiques o la flexibilitat.

Codi 5.2	Implementació del sistema de decisió
Durada	14 dies
Descripció	Implementació del sistema de decisió.

Codi 5.3	Proves del sistema de decisió
Durada	3 dies
Descripció	Definició d'un conjunt de proves, simulació i adequació de paràmetres.

Codi 7.1	Interfície
Durada	4 dies
Descripció	Disseny d'una interfície en Matlab com a punt d'entrada a la monitorització del so respiratori provinent d'un arxiu d'àudio.

Codi 7.2	Implementació
Durada	8 dies
Descripció	Implementació de la interfície.

Codi 7.3	Proves
Durada	3 dies
Descripció	Realització de proves finals amb visualització dels resultats de la identificació de les fases de respiració detectades en un arxiu d'àudio a mesura que el va processant.

Codi 8.1	Documentació (memòria)
Durada	57 dies
Descripció	Elaboració de la memòria del treball.

Codi 8.1	Documentació (presentació virtual)
Durada	4 dies
Descripció	Elaboració de la presentació virtual.

La taula de fites de les activitats és:

Taula 1. Taula de fites del TFM

#	Activitat	Dies	Inici	Fi
1	**Preparació TFM**	**18**	**22/09/14**	**15/10/14**
1.1	Definició del TFM	11	22/09/14	06/10/14
1.2	Elaboració Pla de Treball	7	07/10/14	15/10/14
1.3	Lliurament Pla de Treball		15/10/14	
2	**Definició del marc de treball**	**4**	**16/10/14**	**21/10/14**
3	**Anàlisi del so respiratori**	**16**	**22/10/14**	**12/11/14**
3.1	Enregistrament i filtratge	5	22/10/14	28/10/14
3.2	Estudi i selecció d'atributs	9	29/10/14	10/11/14
3.3	Atributs seleccionats		10/11/14	
3.4	Generació de repositori	2	11/11/14	12/11/14
4	**Lliurament Seguiment 1**		**12/11/14**	
5	**Sistema de decisió**	**22**	**13/11/14**	**12/12/14**
5.1	Selecció	5	13/11/14	19/11/14
5.2	Implementació	14	20/11/14	09/12/14
5.3	Proves	3	10/12/14	12/12/14
6	**Lliurament Seguiment 2**		**12/12/14**	
7	**Interfície**	**15**	**15/12/14**	**02/01/15**
7.1	Disseny	4	15/12/14	18/12/14
7.2	Implementació	8	19/12/14	30/12/14
7.3	Proves	3	31/12/14	02/01/15
8	**Documentació**	**62**	**16/10/14**	**10/01/15**
8.1	Elaboració memòria	57	16/10/14	02/01/15
8.2.	**Lliurament memòria**		**03/01/15**	
8.1	Elaboració presentació	4	05/01/14	08/01/15
8.2.	**Lliurament presentació**		**10/01/15**	

El diagrama Gantt del TFM, que mostra la cronologia i dependència entre activitats, és:

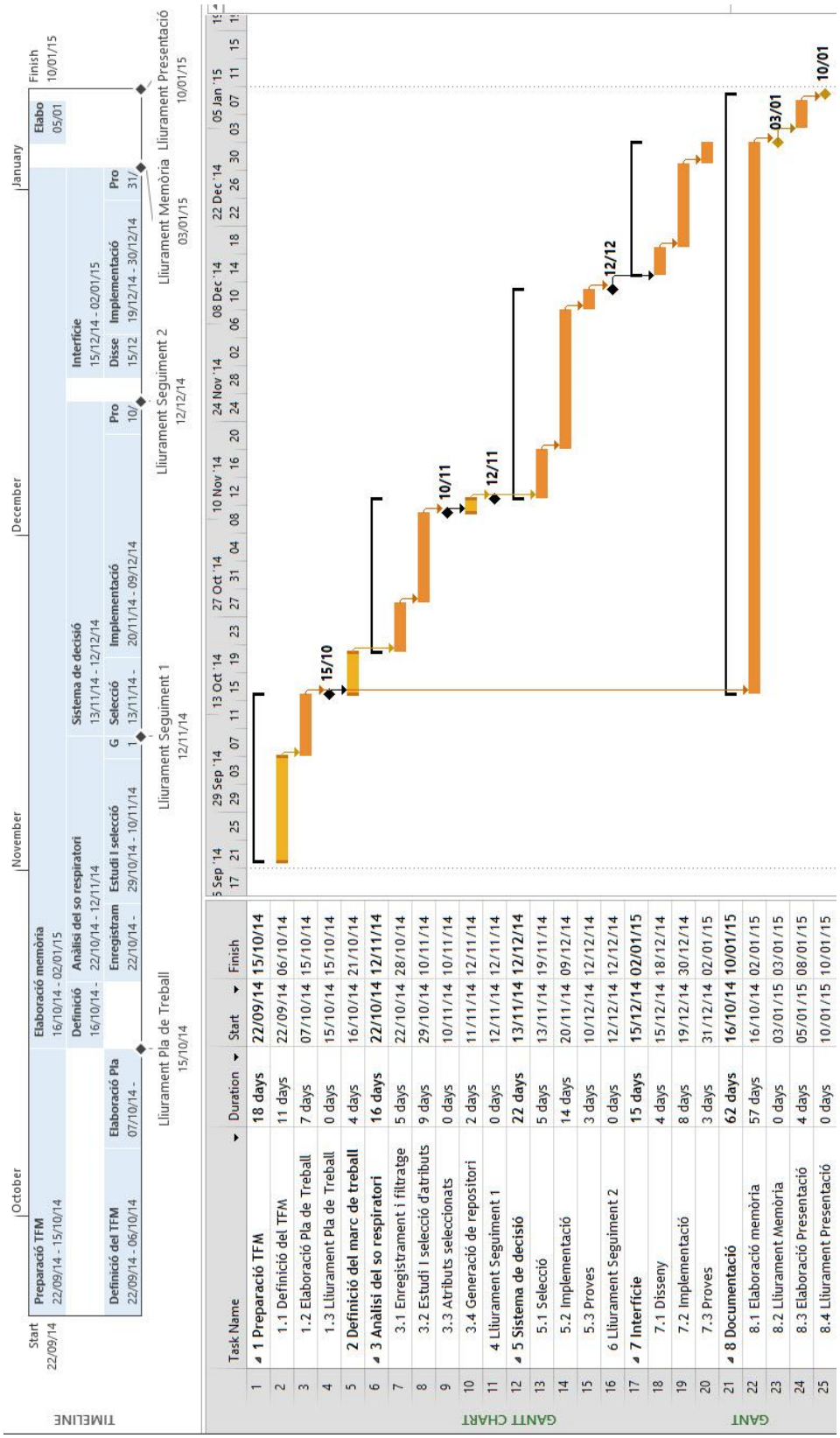


Figura 1. Diagrama Gantt del TFM

1.5. ANÀLISI DE RISCOS

El Treball està suficientment acotat en l'abast com per a què el risc d'incompliment en la data de lliurament (data no prorrogable) sigui pràcticament nul. Ara bé, hi ha dos factors de risc que poden variar el resultat final sense que influeixi de manera rellevant en el conjunt d'eines i coneixement generat. Aquests riscos són:

Risc 1	**Factor de risc: Alt nivell d'errors en la identificació de la respiració per nas o boca**
Grau	mitjà-baix
Mesura de mitigació	Reduir la implementació final a nas o boca i justificar els resultats de la recerca i els inconvenients trobats amb la combinació.

Risc 2	**Factor de risc: Necessitat de més temps en les tasques de codis 3 (anàlisi de la respiració) i/o 5 (Sistema de decisió)**
Grau	mitjà
Mesura de mitigació	Escorçar el temps de la tasca 7 (Interfície) reduint les seves funcionalitats.

1.6. PRODUCTES OBTINGUTS

Com a resultat d'aquest Treball de Fi de Màster, es lliuren els següents productes:

- Una solució desenvolupada en Matlab que identifica en temps real els moviments respiratoris d'inspiració i espiració que procedeixen directament de l'enregistrament a través d'un micròfon.
- Una memòria tècnica que documenta tots els reptes afrontats, experiments i proves realitzades per a l'assoliment de la solució final.
- Una presentació virtual que resumeix els aspectes més rellevants del Treball.

1.7. BREU DESCRIPCIÓ DELS ALTRES CAPÍTOLS

Els següents capítols de la memòria estan estructurats segons l'ordre cronològic de desenvolupament del treball.

El capítol 2, "Antecedents", fa un breu repàs a la literatura existent sobre l'anàlisi de la respiració per computadora. Aquest capítol posa de manifest els diferents escenaris i configuracions inicials definits pels autors les solucions dels quals estan en estreta relació a aquests.

El capítol 3, "Anàlisi del so respiratori", estudia el procés que va des de la captura del so respiratori fins a la generació del model dades necessari per al disseny del sistema d'identificació de moviments respiratoris. Tot això comporta un seguit de tasques encarregades de l'obtenció d'enregistraments respiratoris de voluntaris, l'extracció de característiques acústiques dels enregistraments i la seva avaluació com a atributs diferenciadors dels moviments respiratoris.

El capítol 4, "Sistema d'identificació", proposa un sistema combinat de tècniques per a identificar els moviments respiratoris. Aquest capítol inclou un ampli conjunt de proves amb la finalitat de validar el model de dades i el sistema combinat en diferents configuracions.

El capítol 5, "Interfície d'ús", mostra l'aplicació que permet provar el sistema d'identificació de la inspiració i espiració amb diferents paràmetres de configuració, de manera que permeti avaluar tant la fortalesa de les diferents característiques acústiques i altres paràmetres estudiats en els capítols anteriors. A més, la interfície es desenvolupa en modes consola i entorn gràfic.

El capítol 6, "Conclusions i línies futures de treball", exposa els assoliments aconseguits i les conclusions a les quals s'ha arribat, juntament amb recomanacions sobre vies futures de desenvolupament teòric o pràctic.

Finalment, s'inclou un annex (Annex A) on figura el codi font desenvolupat en Matlab.

CAPÍTOL 2 - ANTECEDENTS

La respiració és el procés fisiològic pel qual els organismes capturen l'oxigen i es desprenen del diòxid de carboni, procés essencial en la vida aeròbica. En l'ésser humà, el sistema respiratori, encarregat de fer possible la respiració, està format pels músculs respiratoris, els pulmons i les vies respiratòries. L'intercanvi de volums d'aire entre l'atmosfera i els pulmons (inspiració) o viceversa (espiració) genera un senyal acústic.

l'auscultació del so respiratori mitjançant l'estetoscopi informa l'especialista de l'estat de les vies respiratòries i de possibles alteracions o malalties. El processament digital del senyal respiratori permet l'especialista avantatges addicionals i complementaris com una monitorització continuada, un diagnòstic objectiu, un diagnòstic remot o diferenciat en el temps, etcètera.

L'aplicació del processament digital del senyal per a l'anàlisi de la respiració no és recent. De la bibliografia seleccionada, la referència més antiga data de l'any 1984 [**1**] i en aquest hi existeix alguna referència a articles de set anys enrere.

El resultat de la cerca realitzada posa de relleu els reptes i diferents plantejaments per abordar l'anàlisi de la respiració humana. Totes les referències trobades es basen en un plantejament basat en el processament posterior del senyal una vegada enregistrat, en el qual estudien formes d'analitzar el senyal per detectar disfuncions o malalties respiratòries tals com l'apnea [**2**, **3**, **4**, **1**], l'asma [**5**, **4**, **1**], l'estrès [**6**, **4**], roncs [**7**, **8**, **9**, **10**, **2**, **3**, **11**] o sibilàncies [**7**, **8**, **4**, **12**, **1**, **13**, **2**].

Per altre banda, un dels majors reptes afrontats està associat al moment de la captura del senyal d'àudio, degut a la poca intensitat del so i el conseqüent problema d'aïllament del soroll i altres fonts d'interferència que embruten el senyal original.

Variades alternatives s'han proposat per afrontar aquest problema. Les opcions presentades es poden dividir entre mètodes invasius i no invasius. Dintre dels invasius es troben l'ús de micròfons especials lligats a la gola a l'alçada de la tràquea [**11**], l'ús de mascaretes amb un micròfon incorporat i altres tipus de sensors adherits al cos. Les opcions no invasives, tot i què són les més còmodes per l'usuari, són les que presenten major interferència de soroll. Altres solucions proposen la utilització de més d'un micròfon [**11**, **14**], micròfons situats

molt a prop de l'usuari [**6**, **10**, **4**] o el control de l'aïllament acústic de la sala d'enregistrament.

Aquesta varietat d'opcions fa que cada proposta triï uns paràmetres freqüencials diferents d'acord a cada escenari.

Es interessant el plantejament pràctic donat en dues referències en les quals es fa servir una arquitectura client-servidor on el client correspon a un dispositiu mòbil mitjançant el qual l'usuari enregistra la seva respiració i s'envia al servidor l'analitza i emet de retorn el diagnòstic [**4**, **6**].

També són variades les tècniques d'aprenentatge computacional (en anglès Machine Learning (ML)) utilitzades en la identificació dels esdeveniments respiratoris. K-veí-més-proper (en anglès k Nearest Neighbour (kNN) [**5**], Naïve Bayes [**10**] , Support Vector Machine [**10**], Random Forest [**10**], Neural Networks [**15**, **9**, **8**] o Hidden Markov Models [**7**] són mecanismes que han sigut utilitzats d'acord als propis criteris dels respectius autors.

CAPÍTOL 3 - ANÀLISI DEL SO RESPIRATORI

3.1. ENREGISTRAMENT I FILTRATGE

3.1.1. Introducció

El so, com també qualsevol altre fenomen que presenti una variació en el temps, es pot representar matemàticament com un senyal.

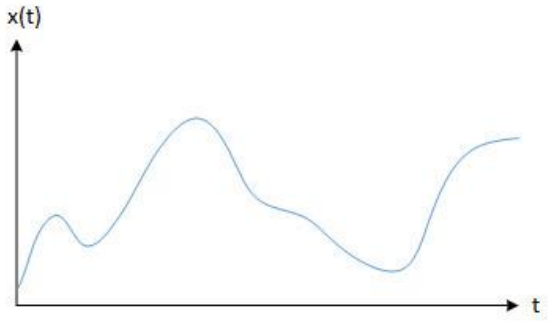


Figura 2. Exemple de senyal analògic

Per a poder fer el processament d'un senyal acústic (senyal analògic) en un ordinador, aquest ha de ser prèviament digitalitzat. La digitalització, consisteix a convertir els valors continus de x i x(t) en valors discrets mitjançant un A/D (Convertidor Analògic Digital). Aquest procés consta de dos passos. Al primer pas, es prendre valors de la variable independent 't' (temps) múltiples d'un període fonamental (T) a través del mostreig. Al segon pas, es realitza la quantificació de variable depenent 'x(t)' (amplitud).

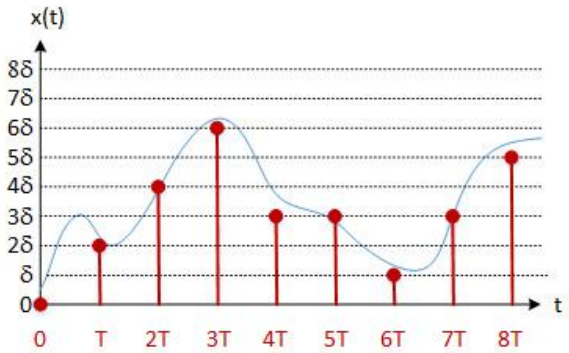


Figura 3. Exemple de senyal analògic i la seva corresponent digitalització

L'elecció dels valors de δ i T condicionaran la qualitat de l'enregistrament del senyal capturat en relació amb l'original. L'elecció de δ afectarà la resolució en els valors de la intensitat del so. L'elecció de T (temps de mostreig) o, equivalentment, la freqüència de mostreig (fs = 1/T) limitarà (pel teorema del mostreig de Nyquist-Shannon) la freqüència més alta que l'enregistrament podrà capturar. Es a dir, si s'escull una freqüència de mostreig de 8.000 Hz, la freqüència màxima que l'enregistrament podrà capturar serà de 8.000/2 = 4.000 Hz. Per tant, si l'espectre audible humà està comprès entre els 20 i 20.000 Hz, el fet d'escollir una freqüència de mostreig superior a 40.000 Hz no aportaria informació addicional a l'enregistrament.

3.1.2. Escenari d'enregistrament

L'objectiu de l'enregistrament acústic de la respiració és analitzar-lo per a extreure informació que el caracteritzi i poder crear un model de dades que serveixi per a un posterior sistema de decisió.

Tal com assenyala la bibliografia, l'elecció de les condicions de gravació són variades, adaptades als propòsits i requisits derivats dels recursos i entorn de gravació. En aquest Treball, les característiques de la sala de gravació i recursos emprats són els següents:

Material físic:

- 1 micròfon RODE NT1-A: Es tracta d'un micròfon de tipus direccional (patró de tipus cardioide), capacitiu d'alta sensibilitat (-31.9 dB) i rang dinàmic superior a 132 dB.
- Interfície A/D FireBox de PreSonus.
- Un ordinador portàtil de la família Intel Core i7.

Programari:

- Matlab versió 7.12.0.365 (R2011a): Utilitzat per a l'elaboració del codi font de tota la programació de la solució, funcions d'anàlisi i generació de gràfiques per a aquesta memòria tècnica.
- Adobe Audition versió 4 (CS5.5): Eina auxiliar de tractament digital del so utilitzat per a l'enregistrament i posterior agrupació manual dels sons respiratoris en arxius independents.

Sala d'enregistrament:

- Una habitació tancada aïllada de l'exterior.
- Una butaca on se senten els voluntaris.
- Un micròfon muntat en un trípode situat a un cantó de la butaca. El micròfon està orientat a la part frontal del cap de l'usuari a una distància aproximada de 20 cm.

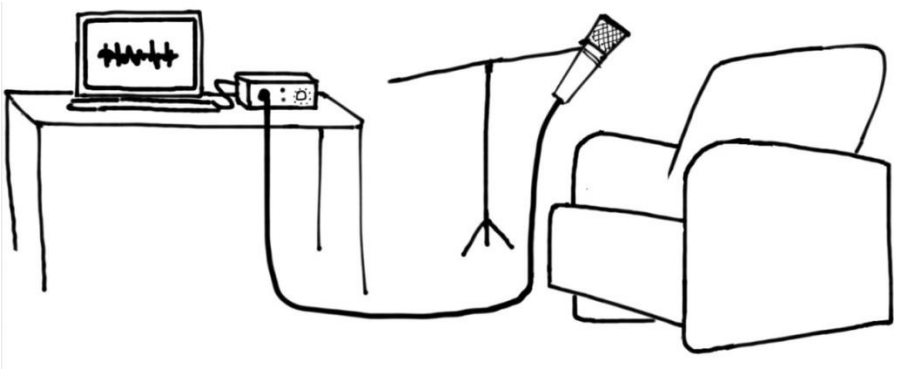


Figura 4. Escenari d'enregistrament

3.1.3. Procés d'enregistrament

Es fa servir com a població quatre voluntaris adults (dos homes i dos dones). Els paràmetres d'enregistrament es fixen amb els següents valors:

- Freqüència de mostreig: 44.100 Hz (per a no perdre informació freqüencial en una primera fase d'avaluació).
- Nombre de canals: 1
- Resolució: 16 bits (alta).

Les gravacions es realitzen a última hora de la nit per reduir el risc d'enregistraments de sorolls externs provinents del carrer o veïnat.

En aquesta primera fase, s'enregistra la respiració bucal i nasal, en sessions independents de 10 minuts, a cada voluntari. Posteriorment, i de manera visual, s'identifiquen els punts d'inici, final i tipus de moviment respiratori (inspiració - espiració) per cada enregistrament. Els resultats gràfics, en el domini temporal i freqüencial, de quatre moviments consecutius en el temps per a cada sessió i voluntari són:

Inspiració i espiració bucals:

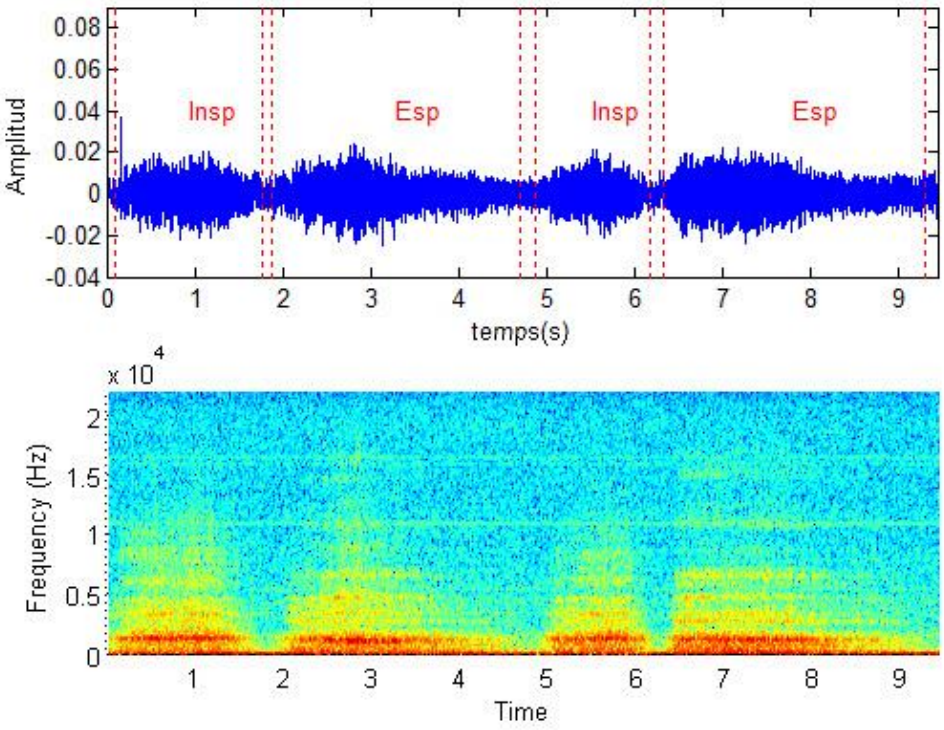


Figura 5. Diagrames temporal i espectral de respiració bucal (voluntari 1)

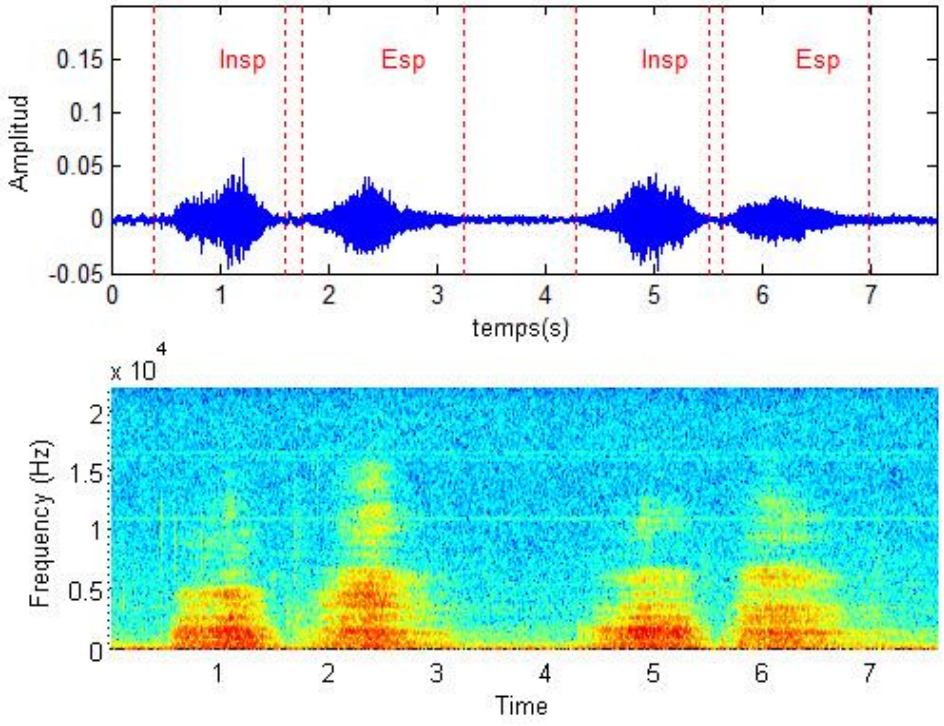


Figura 6. Diagrames temporal i espectral de respiració bucal (voluntari 2)

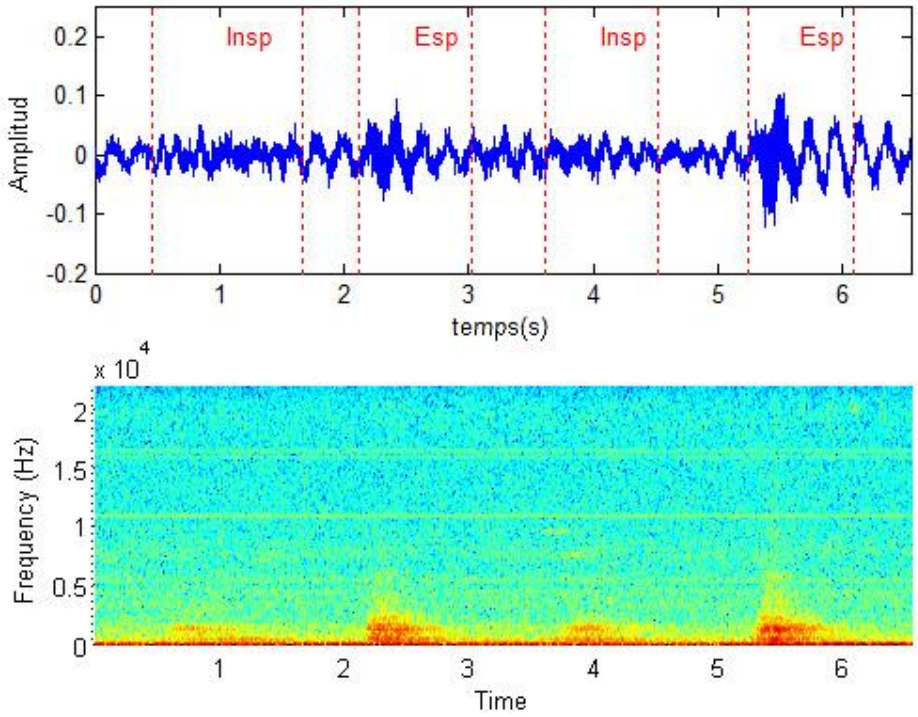


Figura 7. Diagrames temporal i espectral de respiració bucal (voluntari 3)

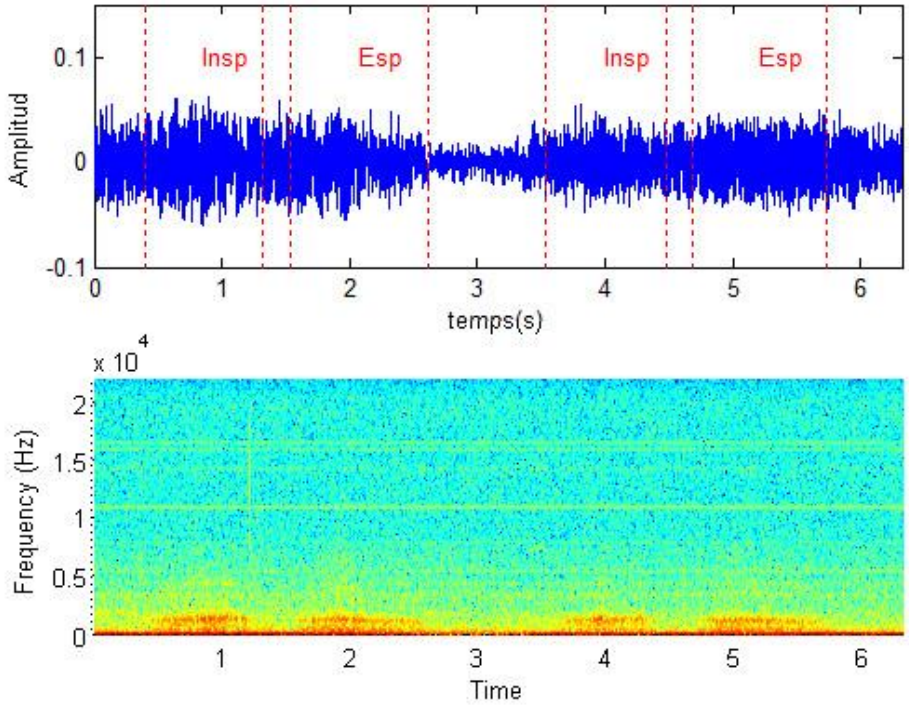


Figura 8. Diagrames temporal i espectral de respiració bucal (voluntari 4)

Inspiració i espiració nasal

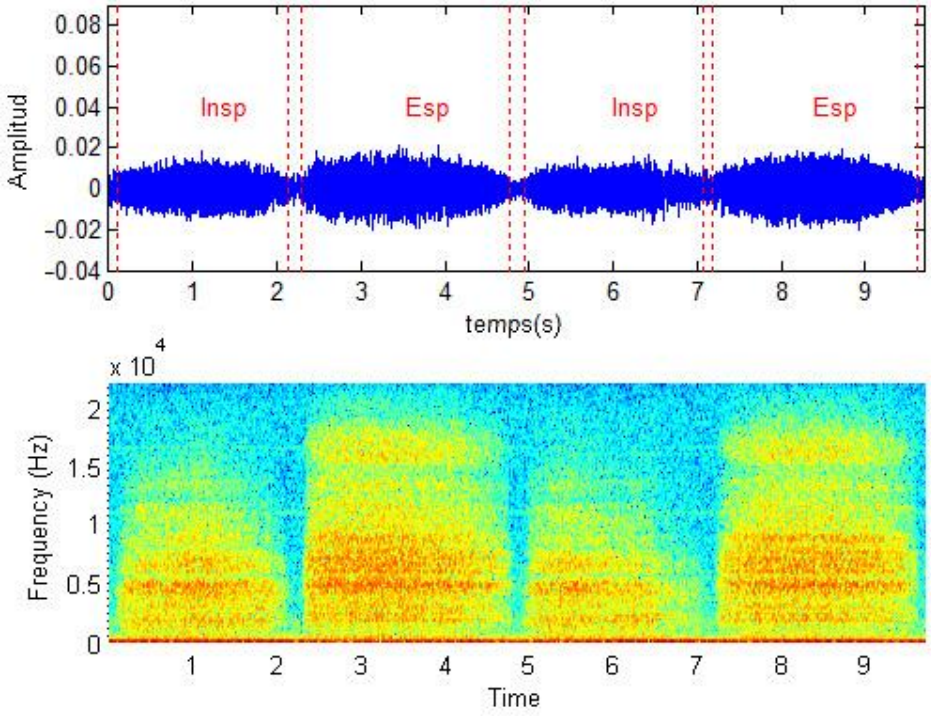


Figura 9. Diagrames temporal i espectral de respiració nasal (voluntari 1)

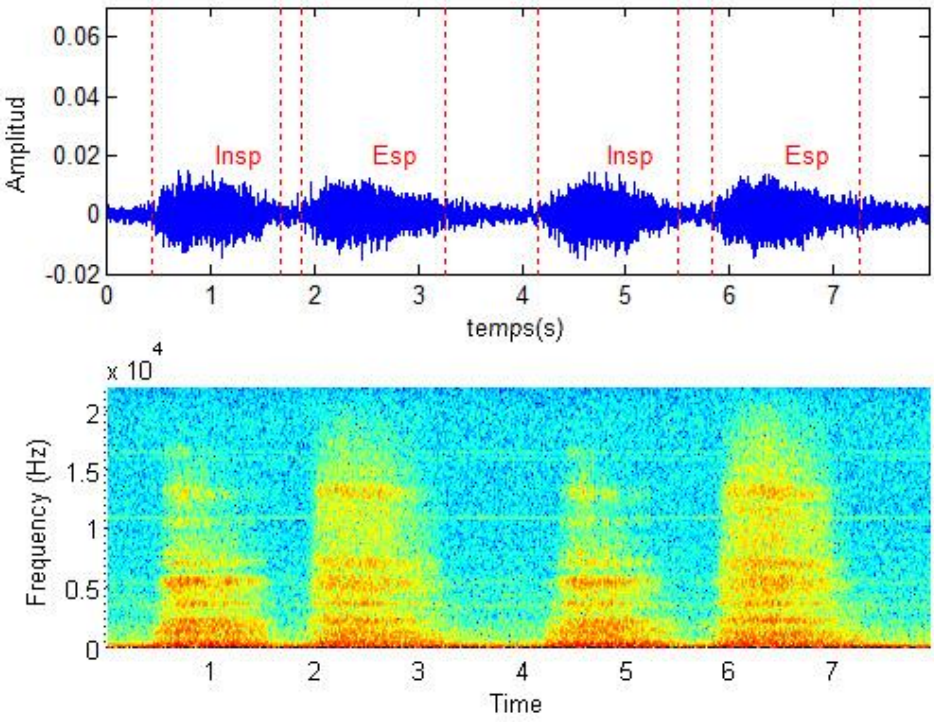


Figura 10. Diagrames temporal i espectral de respiració nasal (voluntari 2)

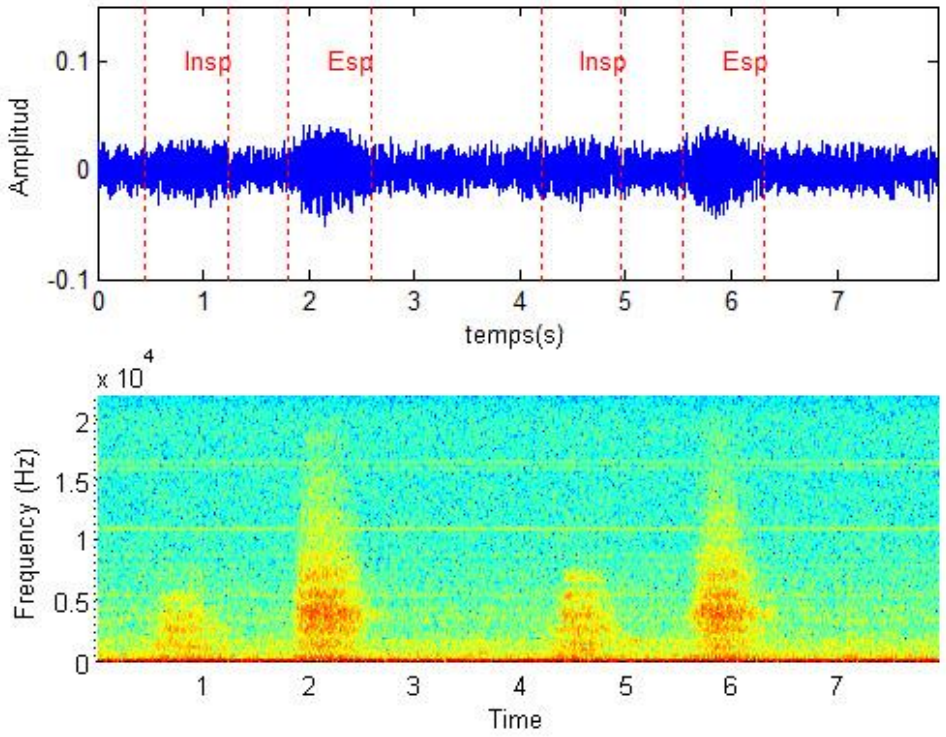


Figura 11. Diagrames temporal i espectral de respiració nasal (voluntari 3)

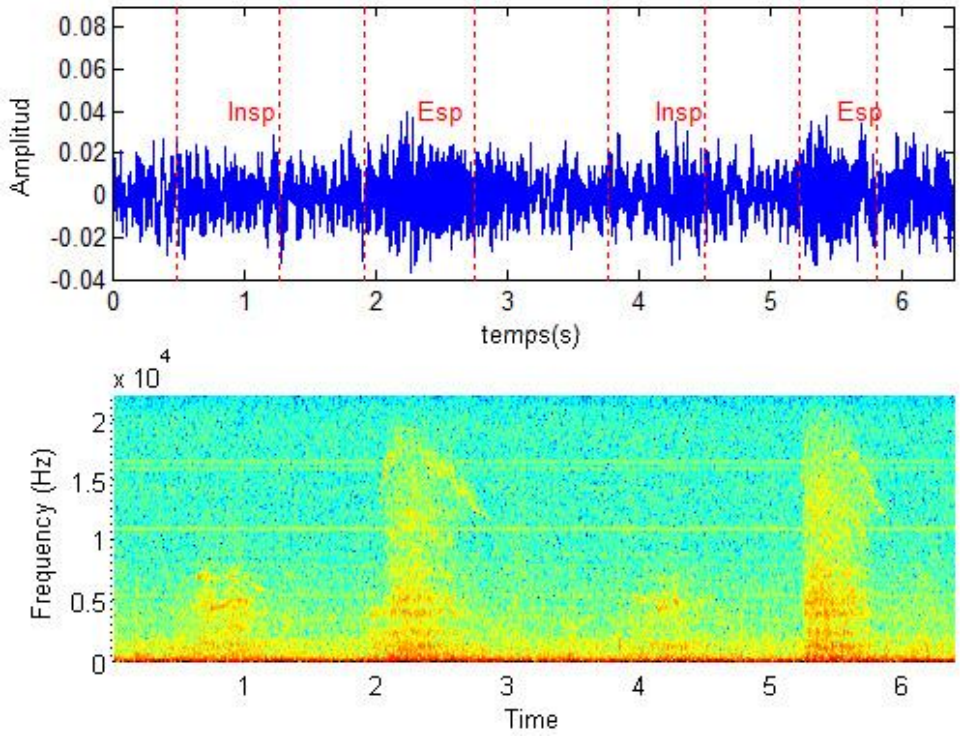


Figura 12. Diagrames temporal i espectral de respiració nasal (voluntari 4)

Una primera anàlisi visual dels diagrames permet assenyalar que:

- Els diagrames temporals mostren que la intensitat de la respiració varia per a un mateix tipus de moviment respiratori. La raó és senzilla: diferents persones poden tenir distintes intensitats de respiració i aquesta pot canviar en el temps. A més, la sensibilitat del micròfon i la distància d'aquest al subjecte afecta directament a aquesta mesura.

  Així doncs, una mesura de la intensitat del senyal sembla que no donarà a priori massa informació sobre el tipus de moviment respiratori. No obstant això, sí que pot ser interessant com una mesura per a la detecció de la presència o no de moviment respiratori. La intensitat servirà per marcar un llindar per sota del qual es considerarà informació no audible o transició entre moviments respiratoris.

- Tant els diagrames temporals com els freqüencials mostren que hi ha un component de soroll constant en tots els enregistraments. Es veu molt clarament com les fases de transició no són nul·les en el temps. Fins i tot, aquest soroll, amb una forta component de baixes freqüències, ressalta més com ona principal a la Figura 7.

  En l'àmbit freqüencial, aquest soroll és present a tot l'espectre però en més intensitat per sota dels 100 Hz aproximadament. Aquest component de soroll per sota de 100 Hz s'ha de treure per a aconseguir una millor relació senyal/soroll que permetrà major sensibilitat, sobretot, quan els moviments respiratoris són de poca intensitat sonora.

- La respiració bucal concentra la majoria de la seva intensitat en freqüències per sota dels 5.000 Hz mentre que la intensitat de la respiració nasal és més homogènia en freqüències, arribant per sobre del 10.000 Hz en el cas de l'espiració.

D'acord a això i, es pot definir de nou els paràmetres d'enregistrament amb els següents valors:

- Freqüència de mostreig: 22.050 Hz.
- Nombre de canals: 1
- Resolució: 16 bits (alta).

La velocitat de mostreig de 22.050 Hz, evitarà que s'enregistrin les freqüències per sobre dels 11.025 Hz i es reduirà el nombre de mostres per segon a la meitat.

Així mateix, en el procés de desenvolupament de la solució, aquesta freqüència s'assigna com un paràmetre general de configuració que servirà per a avaluar, en funció de velocitats de mostreig més reduïdes (16.000 Hz, 11.025 Hz i 8.000 Hz) el cost computacional (es redueixen el nombre de mostres a tractar) i el nivell de qualitat de la solució (a causa de la pèrdua d'informació). Aquest paràmetre, rep el nom de `ASBS_SAMPLING` en el programari desenvolupat.

En els enregistraments ja realitzats per a la creació del model de dades, la reducció de la freqüència de mostreig s'aplica directament als àudios enregistrats a través del programa Audition.

3.1.4. Filtratge

S'ha vist com hi ha un component important de soroll en la banda per sota dels 100 Hz. A la Figura 13 es pot veure en més detall. Es important reduir aquest soroll per augmentar la relació senyal/soroll.

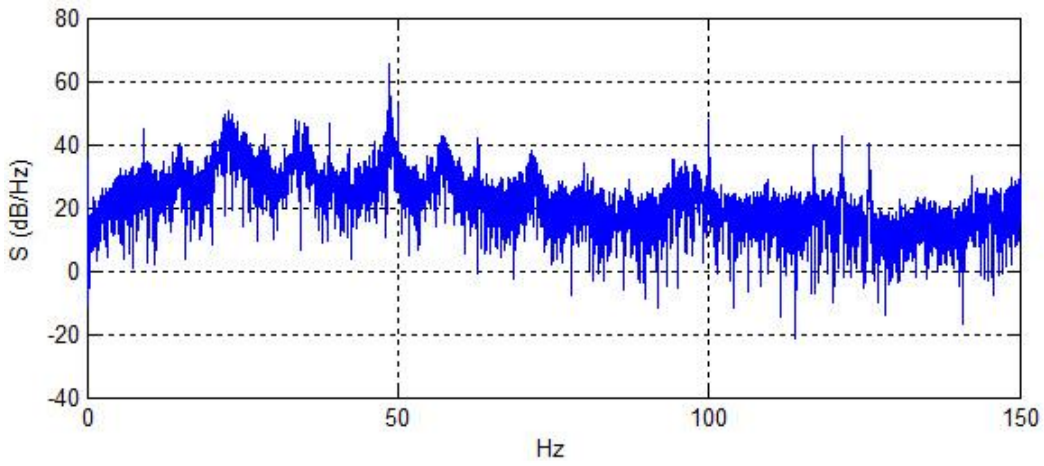


Figura 13. Exemple d'espectre de potència de soroll enregistrat

Per a l'eliminació d'aquestes freqüències es fa servir un filtre el·líptic. Un filtre el·líptic té la propietat de què la caiguda és molt brusca en la banda de transició en comparació amb altres filtres amb un mateix nombre de coeficients (ordre del filtre).

Els paràmetres del filtre el·líptic dissenyat són els següents:

- Ordre 6
- 0.1 dB d'arrissament en la banda de pas.
- 50 dB de caiguda en la banda de transició.

L'obtenció dels zeros (z) i pols (p) de la funció de transferència del filtre s'obté amb el Matlab:

$$[z\ p] = ellip(6, 0.1, 50, 0.00907, 'high') \quad (1)$$

El valor 0.00907 expressa la freqüència de tall en radians i normalitzada:

$$fcn = \frac{\frac{2\pi fc}{fs}}{\pi} = \frac{\frac{2\pi 100}{22050}}{\pi} = 0.00907 \quad (2)$$

Els valors de z i p, com són en funció de la freqüència de mostreig, es parametritzen en funció d'aquesta i s'emmagatzemen en les variables `ASBS_ELLIP_ZEROS` i `ASBS_ELLIP_POLES`.

La representació gràfica de la resposta en freqüència del filtre és:

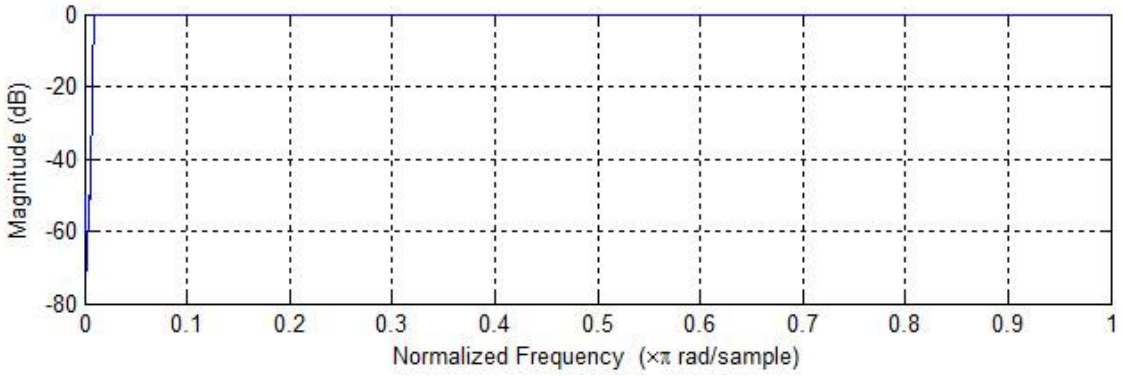


Figura 14. Resposta en freqüència del filtre el·líptic dissenyat

Una mostra resultant de l'aplicació del nou mostreig i amb l'aplicació del filtre és la següent:

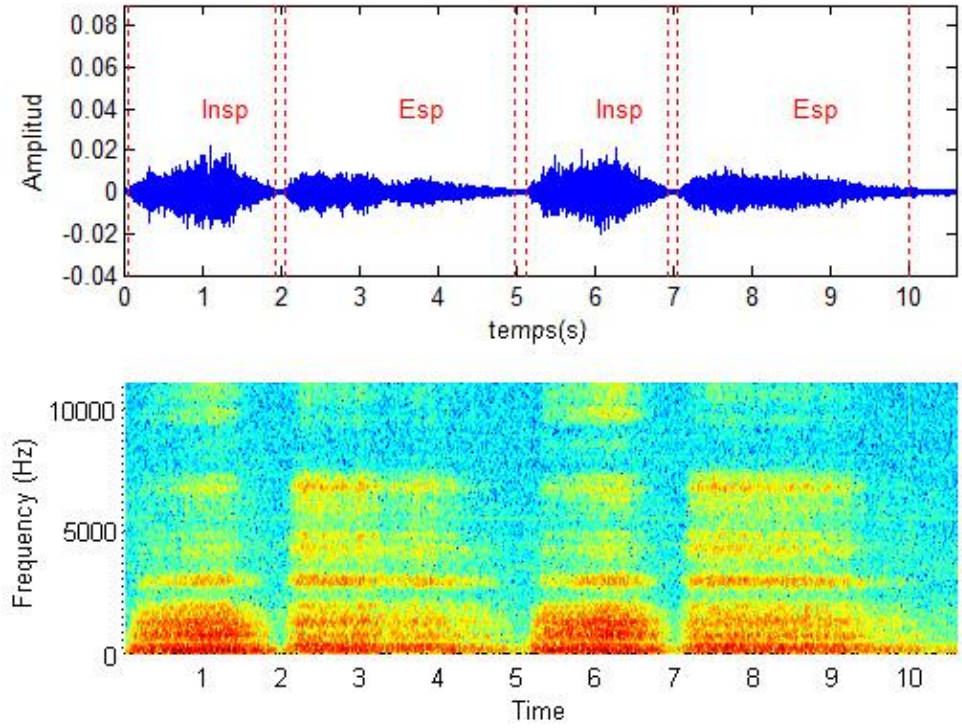


Figura 15. Diagrames temporal i espectral una vegada aplicat el filtre (voluntari 1)

El diagrama temporal mostra com queden més perfilats els moviments respiratoris en relació a abans que no estaven prou definits. En l'àmbit freqüencial, si es compara l'espectrograma abans i després del filtratge, es pot veure l'atenuació.

La següent figura mostra una zona ampliada de l'espectrograma:

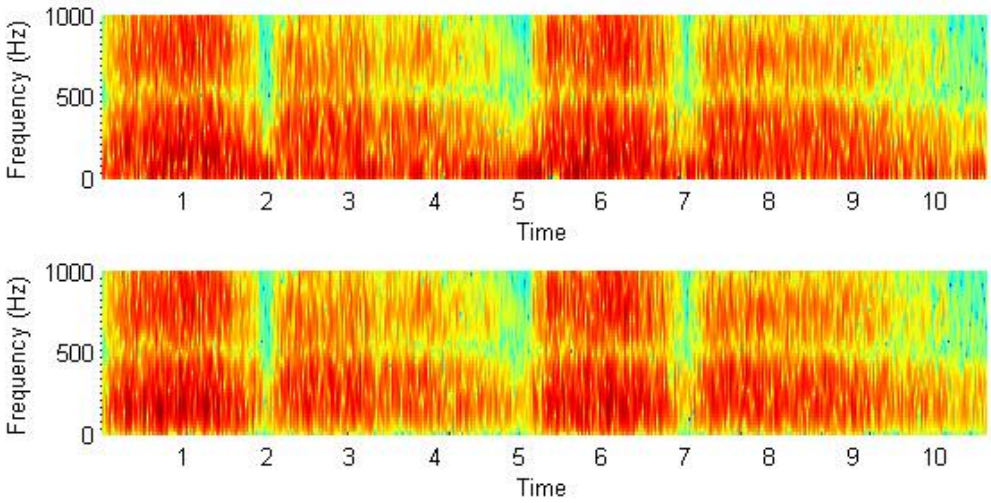


Figura 16. Detall de l'efecte del filtre

3.2. EXTRACCIÓ DE CARACTERÍSTIQUES

3.2.1. FFT

La representació en el domini freqüencial d'una funció continua en el temps es denota per la Transformada de Fourier (TF) i es defineix com:

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-j2\pi f t} dt \tag{3}$$

L'equivalent en els senyals discrets, que són els que manipulen els ordinadors, és la Transformada Discreta de Fourier (DFT) que es defineix com:

$$X[k] = \sum_{0}^{N-1} x[n] e^{-j\frac{2\pi n}{N}k} \qquad k = 0, 1, \ldots, N-1 \tag{4}$$

La FFT (*Fast Fourier Transformation*), és un algorisme de càlcul molt ràpid de la DFT. Mentre que l'ordre de complexitat de la DFT és $O(N^2)$, el de la FFT és de $O(Nlog(N))$.

La FFT és bàsica per al càlcul i extracció d'atributs en el domini freqüencial del so. A l'hora de fer ús de la FFT, aquesta té la limitació en què el nombre de mostres sobre les quals s'aplica ha de ser potència de 2.

3.2.2. Segmentació i encavalcament

L'extracció d'atributs de l'enregistrament es fa mitjançant la segmentació de l'àudio en petits blocs espaiats uniformement en el temps a intervals regulars. Aquests blocs es tracten individualment, de forma que poden caracteritzar la seva estructura. La longitud del bloc es mesura en milisegons i s'anomena finestra. Valors típics utilitzats per la finestra estan compresos entre 30 i 120 ms. Una finestra massa gran requerirà el processament de menys blocs, però possiblement es perdrà informació estructural del senyal.

L'espaiat entre blocs pot ser encavalcat. L'efecte és que cada bloc conté part del bloc anterior i pot servir per recuperar la informació atenuada als extrems per l'enfinestrat

(veure 3.2.3). Aquest Treball contempla l'encavalcament com un paràmetre de configuració (variable `ASBS_OVERLAP`).

Els atributs derivats de l'espectre freqüencial es calculen amb la FFT. Per optimitzar el temps de computació, és adient escollir una longitud de finestra que derivi en un nombre de mostres proper a la següent potència de 2.

Per exemple, una finestra de 100 ms amb una freqüència de mostreig de 22.050 Hz equival a 2.205 mostres per segment i que, per a la FFT, s'hauran d'afegir 1.891 zeros per arribar a la següent potència de 2 més propera (4.096). Si en comptes de 100 ms s'escull una finestra de 92 ms, el nombre de zeros que s'afegirà pel càlcul de la FFT serà de només 20.

Tot i que el valor de la longitud de la finestra pot ser escollit lliurament, la següent taula mostra per a diferents freqüències de mostreig, les longituds de les finestres proposades, el nombre de mostres per bloc i els punts de la FFT:

Taula 2. Propostes de longitud de finestra en funció de la freqüència de mostreig

freq. de mostreig	longitud finestra	# mostres per finestra	Punts FFT
22.050 Hz	92 ms	2.028	2.048
	69 ms	1.521	2.048
	46 ms	1.014	1.024
	23 ms	507	512
16.000 Hz	128 ms	2.048	2.048
	96 ms	1.536	2.048
	64 ms	1.024	1.024
	32 ms	512	512
11.025 Hz	92 ms	1.014	1.024
	69 ms	760	1.024
	46 ms	507	512
	23 ms	253	256

8.000 Hz	128 ms	1024	1.024
	96 ms	768	1.024
	64 ms	512	512
	32 ms	256	256

3.2.3. Enfinestrat

El procés de segmentació, suposa extreure del conjunt, una part petita del senyal. Això crea en la Transformada de Fourier distorsions degudes al caràcter no periòdic del senyal en la curta longitud del segment. Per a solucionar aquest efecte negatiu, s'aplica al segment, prèviament a la Transformada de Fourier, una funció finestra. Aquesta funció suavitza els extrems de l'interval, fent que aquests tendeixin a zero i, al mateix temps, emfatitzant la part central. Així, s'accentuen les propietats característiques del segment.

Les funcions finestres més comunes que s'apliquen en el tractament acústic són les finestres de Hamming i Hanning. Són finestres molt similars on només canvien els coeficients. En aquest treball s'aplica l'enfinestrat de Hamming, ja que és la que s'utilitza comunament per a l'extracció dels coeficients MFCC (pàg. 35).

La finestra de Hamming està definida per la següent expressió:

$$w(n) = 0.54 - 0.46\cos\left(2\pi\frac{n}{N}\right), \qquad 0 \leq n \leq N \tag{5}$$

L'aplicació de la finestra consisteix en multiplicar el segment per la funció finestra:

$$xh(n) = x(n) \cdot w(n), \qquad 0 \leq n \leq N \tag{6}$$

El següent diagrama, mostra un exemple de la distorsió produïda en la FFT i l'efecte corrector de l'enfinestrat sobre un segment de 125 mostres d'un senyal sinusoïdal de 100 Hz mostrejat a 1.000 Hz:

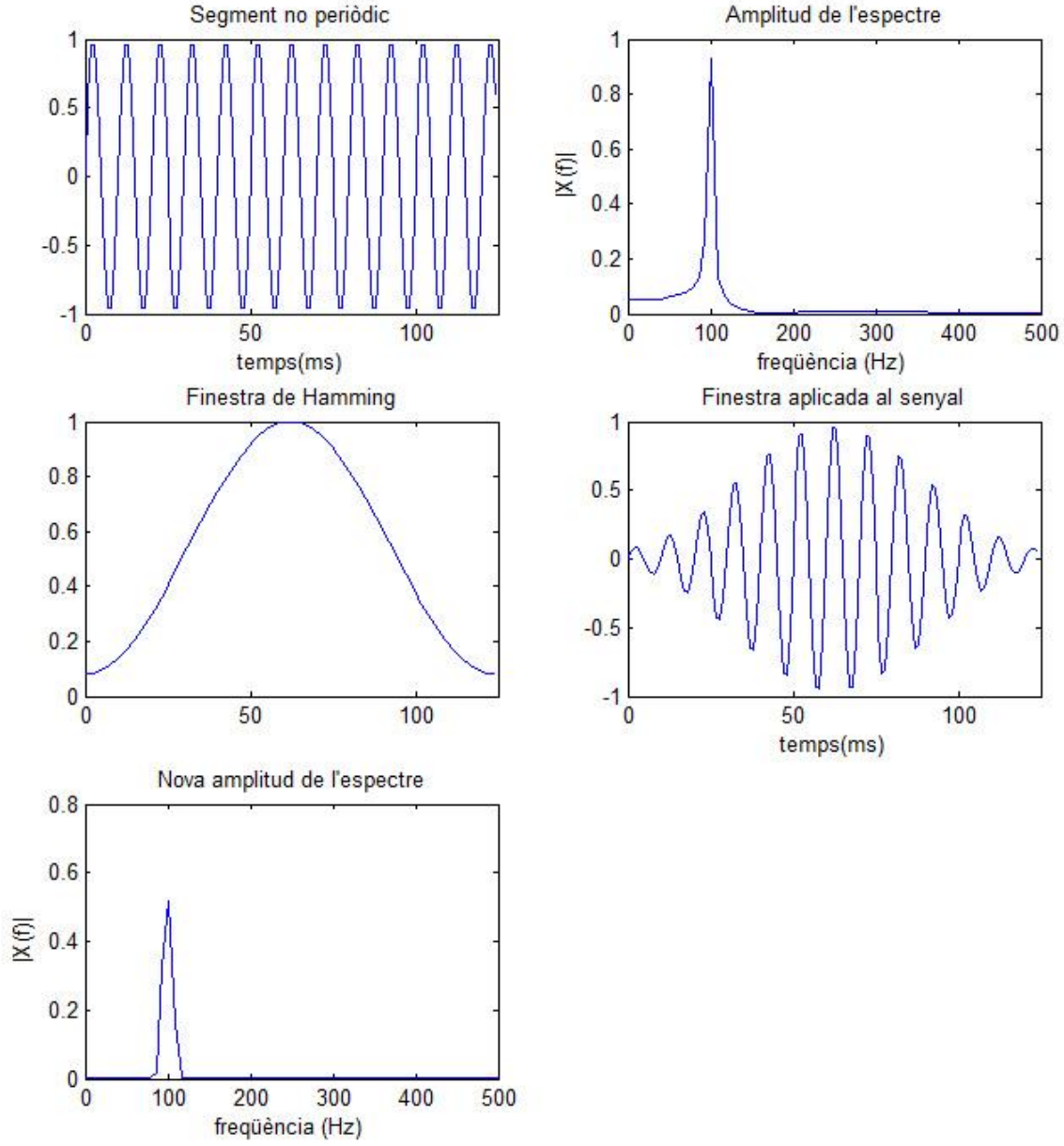


Figura 17. Enfinestrat Hamming sobre un senyal mostra i efecte resultant

3.2.4. Preparació

Per l'estudi dels diferents atributs que poden extreure-se'n de les mostres de respiració, s'han d'identificar prèviament i de forma manual els moviments respiratoris. Aquest és un procés tediós la finalitat del qual és permetre l'estudi dels atributs i la creació del model de dades o repositori.

Per a dur-lo a terme, es fa servir el programari d'edició d'àudio Audition. Es concatenen tots els enregistraments dels voluntaris en un únic arxiu (de nom breathing.wav). Posteriorment, i de manera manual, s'identifiquen visualment els punts de separació entre moviments respiratoris i es registren aquestes marques de temps en un arxiu Excel (breathing.xlsx) de dues columnes. La primera columna inclou la marca de temps i la segona columna inclou el tipus de moviment respiratori produït just abans d'aquesta marca (1 = inspiració bucal, 2 = espiració bucal, 3 = inspiració nasal, 4 = espiració nasal).

El resultat d'aquesta tasca és un arxiu d'àudio de 48 minuts i 53 segons de duració (breathing.wav), constituïts per un total de 1.784 moviments respiratoris perfectament identificats en la seva posició temporal per l'arxiu breathing.xlsx. Aquests arxius són la base sobre la qual s'extreuen les característiques per a construir el repositori.

Les versions de l'arxiu d'àudio breathing.wav a les freqüències de mostreig de la Taula 2 i el filtratge previ es realitza en el programari desenvolupat amb les funcions `asbsdownsampling` i `asbsfilter`.

3.2.5. Característiques

Les característiques són atributs numèrics que es treuen del segment del senyal que serveixen per a descriure la seva estructura. Hi ha característiques que es treuen del domini temporal del senyal i altres que es treuen del seu domini freqüencial. La finalitat és permetre distingir els quatre moviments respiratoris diferents objectius d'aquest Treball: inspiració bucal, espiració bucal, inspiració nasal i espiració nasal.

Per això, a partir dels moviments respiratoris i les seves marques de temps emmagatzemats en breathing.wav i breathing.xlsx, s'extreuen les característiques que a continuació es detallen, mostrant la seva "fortalesa" com a paràmetre de distinció entre moviments respiratoris.

Al programari desenvolupat, aquesta tasca està a càrrec de les funcions `asbsframefeatures` i `asbsfeatures`.

Mitja quadràtica (RMS)

La mitja quadràtica (de l'anglès, *Root Mean Square (RMS)*) és una mesura estadística de magnitud calculada directament sobre les mostres del segment del senyal.

La seva expressió matemàtica és:

$$RMS = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x[n]^2} \tag{7}$$

L'RMS és indicador de la intensitat del senyal i, en aquest Treball, es fa servir per a definir el llindar per sota del qual, la informació del segment s'ha de considerar no audible. Aquests segments que no arriben a aquest llindar correspondrien a les fases de transició entre moviments respiratoris, tal com es veia en les il·lustracions de l'apartat 3.1.3. El valor del llindar, es defineix a partir de l'experimentació, en funció de l'escenari i proves realitzades.

De la fase d'enregistrament, i una vegada filtrat el senyal (apartat 3.1.4.) el senyal residual i l'energia de cadascú dels segments que resta en un enregistrament sense respiració representatiu de 30 segons és:

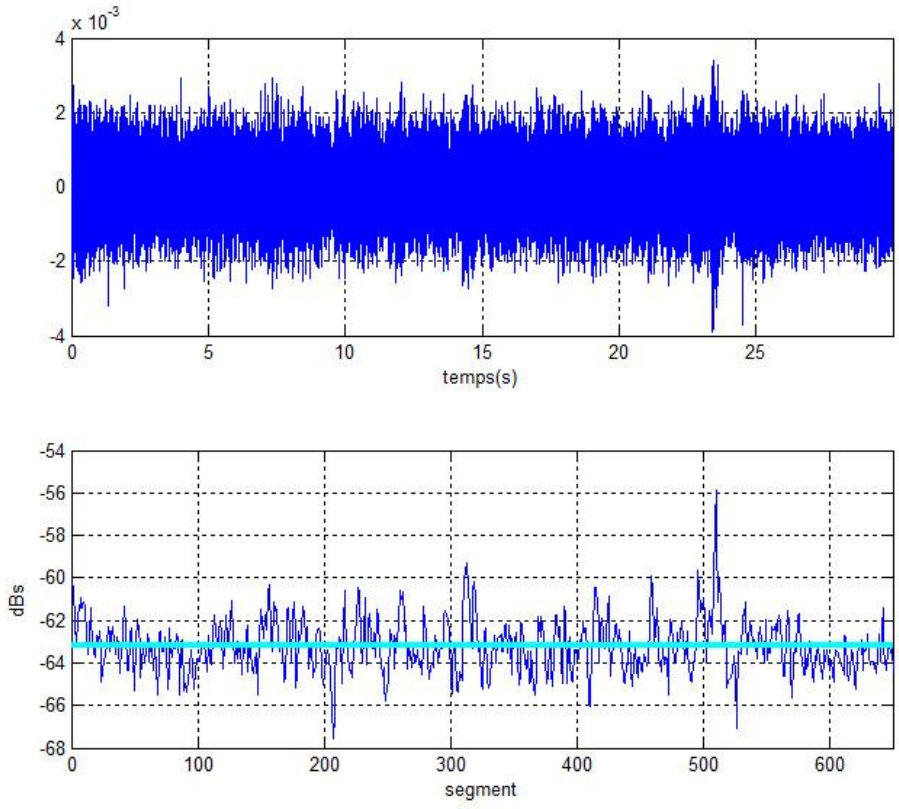


Figura 18. Energia de llindar

De la figura es desprèn que l'energia màxima (en relació a amplitud 0) dels segments de senyal que presenten informació només associable a soroll, està compresa majoritàriament entre els -62 dB i els -65 dB, amb un valor mitjà de -63,20 dB. Escollir un llindar de -58 dB, sembla raonable, ja que significa més de 3 dB de caiguda per sobre dels -62 dB. Expressat en lineal, -58 dB equival a 0.00125. Aquest valor de llindar s'emmagatzema en la variable global `ASBS_THRESHOLD`.

S'ha de tenir en compte, que si el soroll ambient o qualsevol altre so no relacionat amb la respiració, no està per sota del llindar establert, influirà negativament en la qualitat d'identificació de moviments respiratoris.

Aquest paràmetre s'identifica en el programari desenvolupat com `ASBS_RMS`. El càlcul de la característica s'implementa en la funció `asbsrms`.

Centroide espectral (SC)

El centroide espectral (en anglès, *Spectrum Centroid (SC)*), equival al centre de massa o punt d'equilibri de la magnitud de l'espectre freqüencial del senyal.

La seva expressió matemàtica és:

$$SC = \frac{\sum_{n=1}^{N}(n \cdot X[n])}{\sum_{n-1}^{N} X[n]} \tag{8}$$

On n és l'índex de la banda de freqüència de l'espectre i X[n] la magnitud de l'espectre a la banda n.

Per a normalitzar el valor del SC i que no sigui dependent de la freqüència de mostreig, es pot fer servir la següent expressió:

$$SCR = \frac{SC}{SR} \tag{9}$$

on SR és la freqüència de mostreig.

En l'àmbit auditiu, el centroide espectral indica l'àrea freqüencial predominant de l'espectre del senyal i es pot interpretar com un nivell de la brillantor del so.

Si s'obté aquesta característica de les mostres agrupades, es pot analitzar la relació existent entre els valors que pren el centroide en funció als diferents moviments respiratoris.

Una manera visual de representar aquesta relació és a través d'una distribució normalitzada del centroide per a cada moviment respiratori de l'arxiu base (breathing.wav).

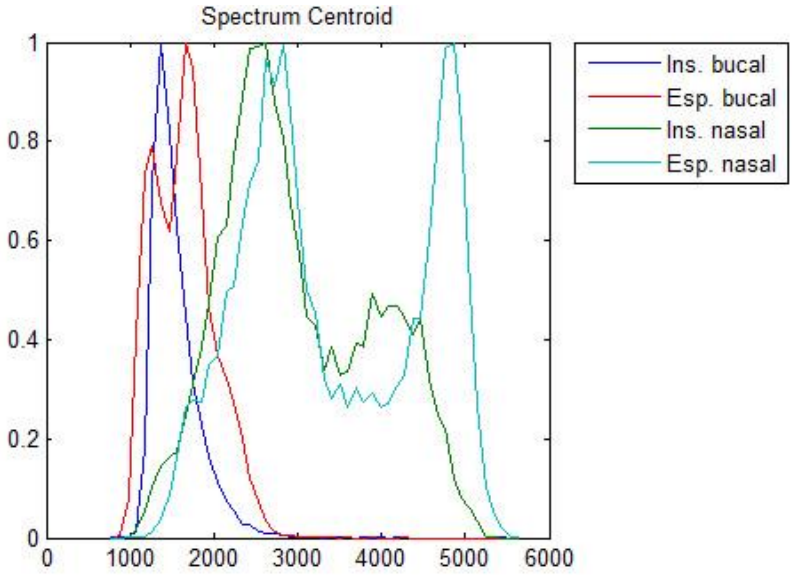


Figura 19. Histograma del *Spectrum Centroid*

De la figura es pot veure com es distribueix el Centroide Espectral en funció del tipus de moviment respiratori. Ressalta com la respiració nasal té més components d'altes freqüències que la respiració bucal i com existeix una sensible diferència entre inspiració i espiració.

Aquesta característica s'identifica en el programari desenvolupat com `ASBS_CENTROID`. El seu càlcul s'implementa en la funció `asbscentroid`.

Plana espectral (SF)

La plana espectral (en anglès, *Spectrum Flatness*), és un indicador del nivell d'homogeneïtat de distribució de la magnitud a les bandes freqüencials i ve a significar si el senyal és de tipus sorollós (un SF proper a 1) o tonal (SF proper a 0).

Es calcula mitjançant el quocient entre la mitjana geomètrica i la mitjana aritmètica de les amplituds de l'espectre freqüencial de senyal.

La seva expressió matemàtica és:

$$SF = \frac{\sqrt[N]{\prod_{n=1}^{N} X[n]}}{\frac{\sum_{k=1}^{K} X[n]}{N}} \tag{10}$$

La distribució normalitzada dels valors que pren aquesta característica per a cada moviment respiratori de l'arxiu base és:

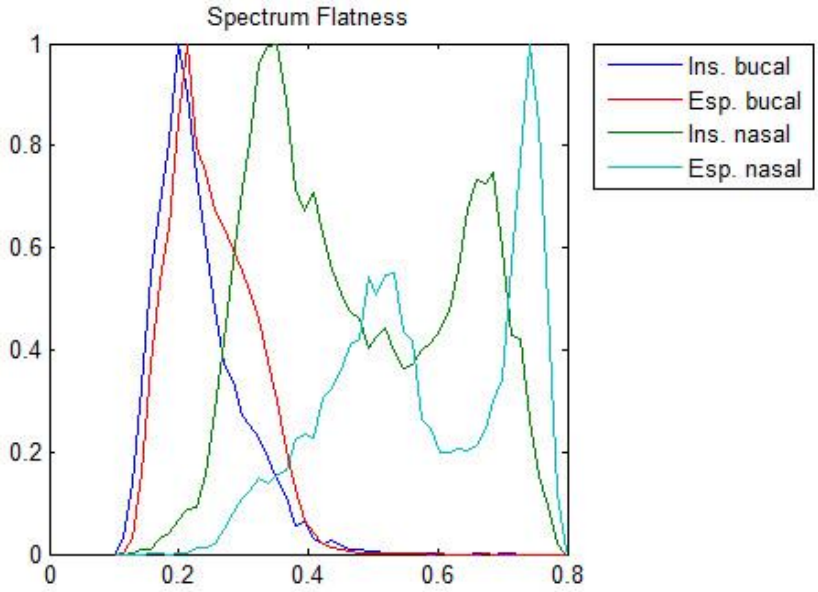


Figura 20. Histograma del *Spectrum Flatness*.

De la figura es desprèn com la respiració nasal fa servir més homogèniament l'espectre freqüencial davant la respiració bucal. Aspecte que ja es va descobrir en l'anàlisi posterior al procés d'enregistrament (capítol 3.1.3.).

Aquesta característica s'identifica en el programari desenvolupat com `ASBS_FLATNESS`. El seu càlcul s'implementa en la funció `asbsflatness`.

Taxa de pas per zero (ZCR)

La taxa de pas per zero (en anglès, *Zero-Crossing Rate* o ZCR) és una característica temporal que indica la taxa per segon de canvi de signe del senyal.

La seva expressió matemàtica és:

$$ZCR = \frac{SR}{N}\sum_{n=1}^{N} |sign(x[n]) - sign(x[n-1])| \quad (11)$$

On SR és la freqüència de mostreig, N el nombre de mostres del segment, x[n] el valor de la mostra i sign() la funció signe definida com:

$$sign(x[n]) = \begin{cases} 1, & x[n] \geq 0 \\ 0, & x[n] < 0 \end{cases} \quad (12)$$

Si es considera la taxa de pas de zero normalitzada pel nombre de mostres, no es té en compte la freqüència de mostreig:

$$ZCR = \frac{1}{N}\sum_{n=1}^{N} |sign(x[n]) - sign(x[n-1])| \quad (13)$$

La taxa de pas per zero, és indicador representatiu de la freqüència dominant, ja que serà aquesta la que contribuirà amb major mesura al canvi de signe en l'ona del senyal.

La distribució normalitzada dels valors que pren aquesta característica per a cada moviment respiratori de l'arxiu base és:

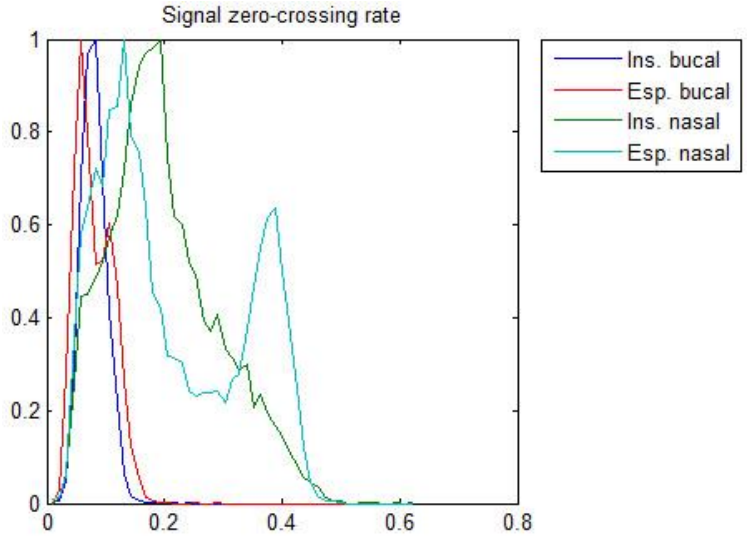


Figura 21. Histograma del *Zero-Crossing Rate*

Aquesta característica s'identifica en el programari desenvolupat com `ASBS_ZCR`. El seu càlcul s'implementa en la funció `asbszcr`.

Atenuació d'espectre (RO)

L'atenuació d'espectre (en anglès, Spectrum Roll-Off (RO)), es defineix com la freqüència sota la qual un determinat percentatge de la distribució de l'espectre del senyal està concentrat.

La seva expressió matemàtica és:

$$RO = \sum_{n=1}^{Nr} X[n] = R \sum_{n=1}^{N} X[n] \tag{14}$$

On Nr és la banda de freqüència de RollOff, X[n] la magnitud de l'espectre a la banda n i R el factor de RollOff. Normalment, al factor R sol assignar-se el valor 0.85, indicant per sota de quina freqüència es concentra la major part de l'energia de l'espectre del senyal.

La distribució normalitzada dels valors que pren aquesta característica per a cada moviment respiratori de l'arxiu base és:

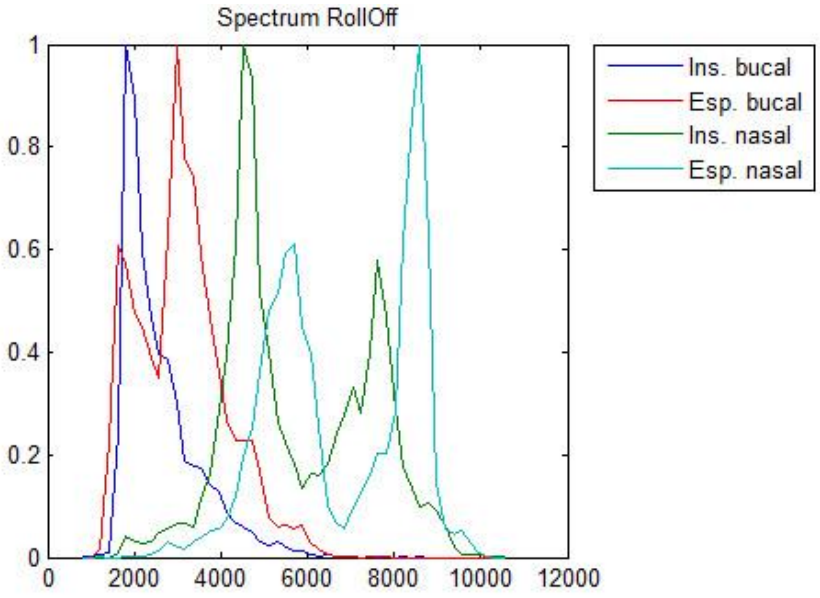


Figura 22. Histograma del *Spectrum RollOff*

Aquesta característica s'identifica en el programari desenvolupat com `ASBS_ROLLOFF`. El seu càlcul s'implementa en la funció `asbsrolloff`.

Ratio RMS en freqüència

Es pot analitzar la relació entre el percentatge d'energia acumulada sobre el total d'energia en funció de la freqüència. Experimentalment i, a partir de les mostres del model, s'obté la següent distribució mitjana:

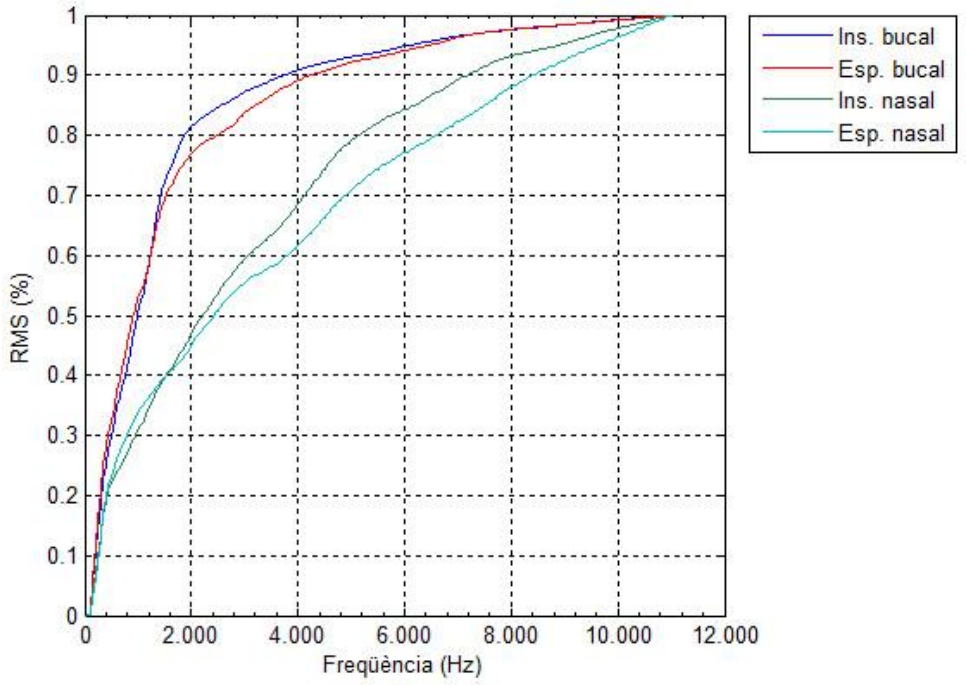


Figura 23. Mitja d'acumulat de magnitud en funció de la freqüència

Per a l'obtenció d'aquesta informació s'ha realitzat una funció específica per a trobar els valors mitjans per a cada banda de 50 Hz d'amplada. La funció implementada per aquest objectiu és la funció `asbsrmsratios`.

De resultes de les dades de la figura, sembla convenient afegir aquesta característica a la freqüència de tall de 2.000 Hz.

La distribució normalitzada dels valors que pren aquesta característica per a cada moviment respiratori de l'arxiu base és:

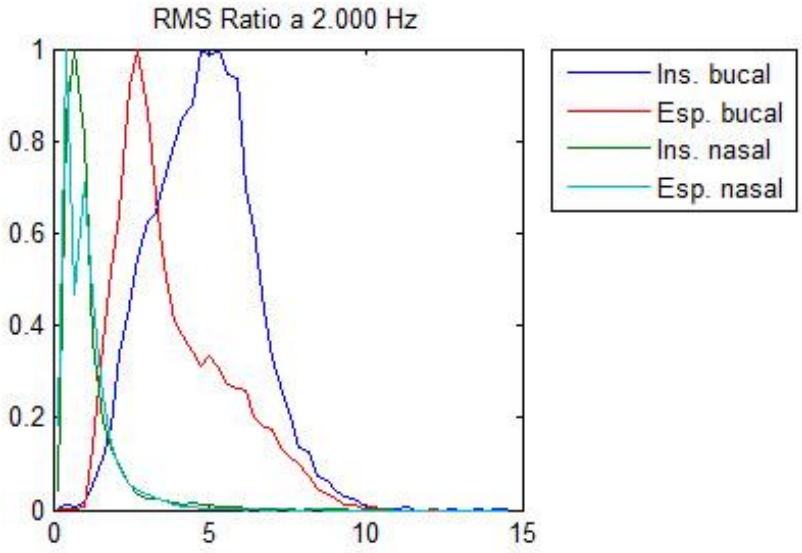


Figura 24. Histograma del ratio energia acumulada a 2.000 Hz

Aquesta característica s'identifica en el programari desenvolupat com `ASBS_RMSRATIO`. El seu càlcul s'implementa en la funció `asbsrmsratio`.

Coeficients cepstrals a les freqüències de Mel (MFCC)

Els Coeficients cepstrals a les freqüències de Mel (en anglès, Mel Frequency Cepstral Coefficients (MFCC)), són de gran aplicació en la parametrització de la veu per al reconeixement de la parla o interlocutor.

Es basen en l'obtenció dels cepstrums del senyal. Un Cepstrum es defineix com la transformada inversa de Fourier del logaritme de l'espectre del senyal:

$$Cepstrum(x[n]) = \mathcal{F}^{-1}(\log(|\mathcal{F}(x[n])|) \quad (15)$$

En el cas dels MFCC, la transformada inversa es substitueix per la transformada discreta de cosinus (DCT) i es fa servir un banc de filtres previ a aplicar sobre l'espectre del senyal. El banc de filtres està basat en l'escala Mel.

L'escala Mel és una representació de la percepció que té l'oïda humana de la distància entre intervals tonals. Aquesta no és constant en tot l'espectre freqüencial. Mentre que existeix linealitat en la percepció d'intervals separats linealment per sota del 500 Hz, per sobre de

500 Hz, freqüències espaiades exponencialment són percebudes com espaiades linealment. L'escala Mel representa aquesta percepció.

L'expressió és:

$$mel(f) = 2595log_{10}(1 + \frac{f}{700}) \quad (16)$$

i la seva funció inversa és:

$$f = 700(10^{\frac{mel}{2595}} - 1) \quad (17)$$

On f és la freqüència en Hz i mel(f) la seva correspondència a l'escala Mel. En la següent gràfica es pot veure la relació:

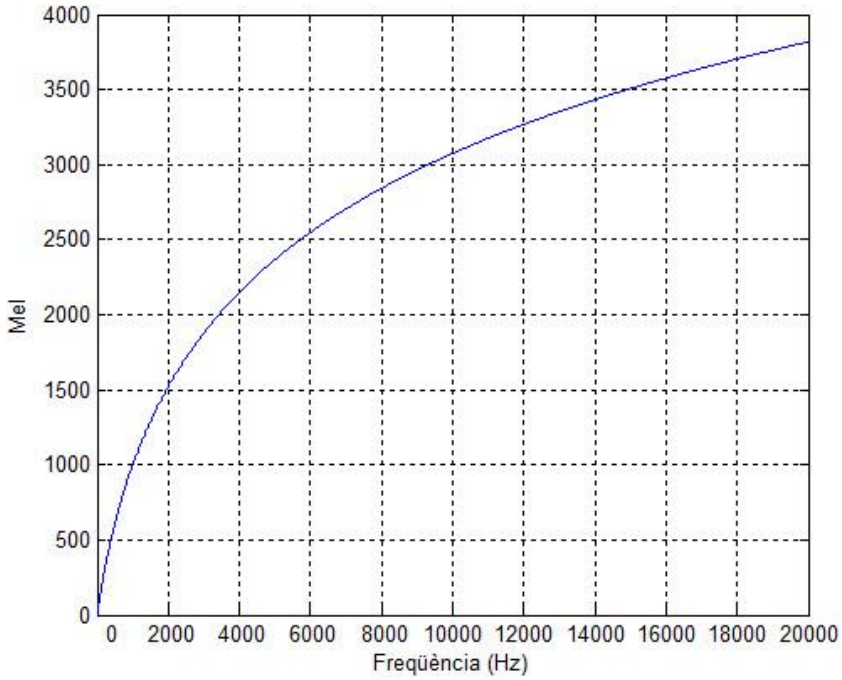


Figura 25. Relació Hz - Mel

El banc de filtres, està constituir per un nombre determinat de filtres triangulars, equidistants d'acord l'escala Mel de freqüències i d'àrea unitat.

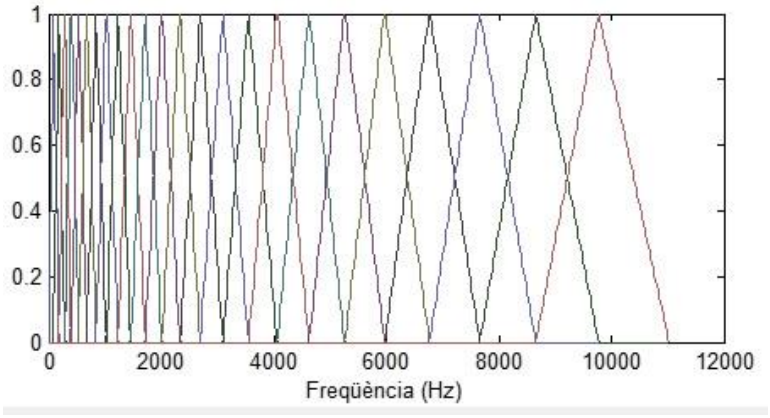


Figura 26. Banc de filtres Mel

El banc de filtres de la Figura 26 està format per 24 filtres equidistants en freqüència segons l'escala mel i considerant una freqüència de mostreig de 22.050 Hz. Les freqüències centrals del banc de filtres són:

Taula 3. Relació de freqüències del banc de filtres

#	Hz Mel	Hz	#	Hz Mel	Hz
1	127	84	13	1.651	2.331
2	254	177	14	1.778	2.693
3	381	282	15	1.905	3.098
4	508	399	16	2.032	3.551
5	635	530	17	2.159	4.058
6	762	677	18	2.286	4.626
7	889	841	19	2.414	5.261
8	1.016	1.025	20	2.541	5.973
9	1.143	1.231	21	2.668	6.769
10	1.270	1.461	22	2.795	7.660
11	1.397	1.719	23	2.922	8.658
12	1.524	2.008	24	3.049	9.775

Així doncs, el procés d'obtenció dels coeficients MFCC té les següents etapes específiques aplicades al segment (una vegada s'ha fet l'enfinestrat):

1. Obtenir el mòdul de l'espectre freqüencial:

$$|X[n]| = |FFT(x[n])| \tag{18}$$

2. Multiplicar-lo pel banc de filtres Mel i sumar l'energia en cada filtre:

$$E[m] = \sum_{n=1}^{N} |X[n]| \cdot H[m], \qquad 1 \leq m \leq M \tag{19}$$

on M és el nombre de filtres del banc I H[m] és el filtre d'índex m.

3. Calcular el logaritme de totes les energies resultants:

$$\log(E[m]), \qquad 1 \leq m \leq M \tag{20}$$

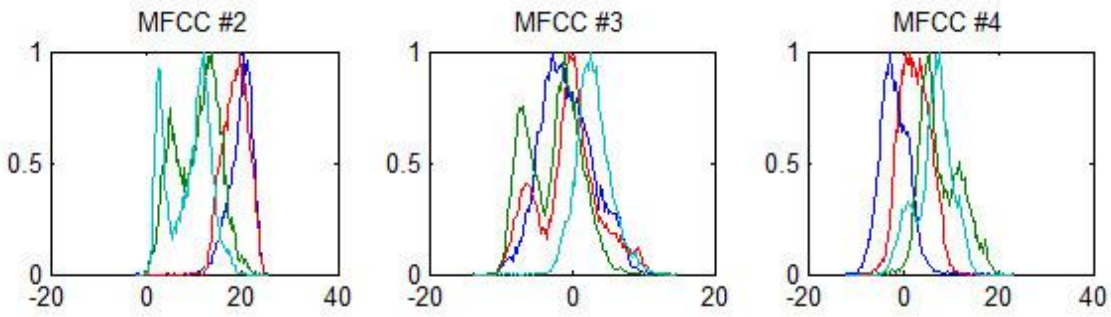
4. Aplicar la Transformada Discreta de Cosinus al logaritme de totes les energies:

$$MFCC[m] = \sqrt{\frac{2}{N}} \sum_{k=1}^{N} \log(E[k]) \cos\left(m\left(k - \frac{1}{2}\right)\frac{\pi}{N}\right), \qquad 1 \leq m \leq F \tag{21}$$

On F és el nombre de filtres del banc.

5. Els coeficients MFCC, són les amplituds de l'aplicació de la DCT. En fan servir els coeficients m = 2..13 i descartar la resta.

Les següents figures mostren les distribucions de les mostres de l'arxiu base per als 12 coeficients MFCC considerats:



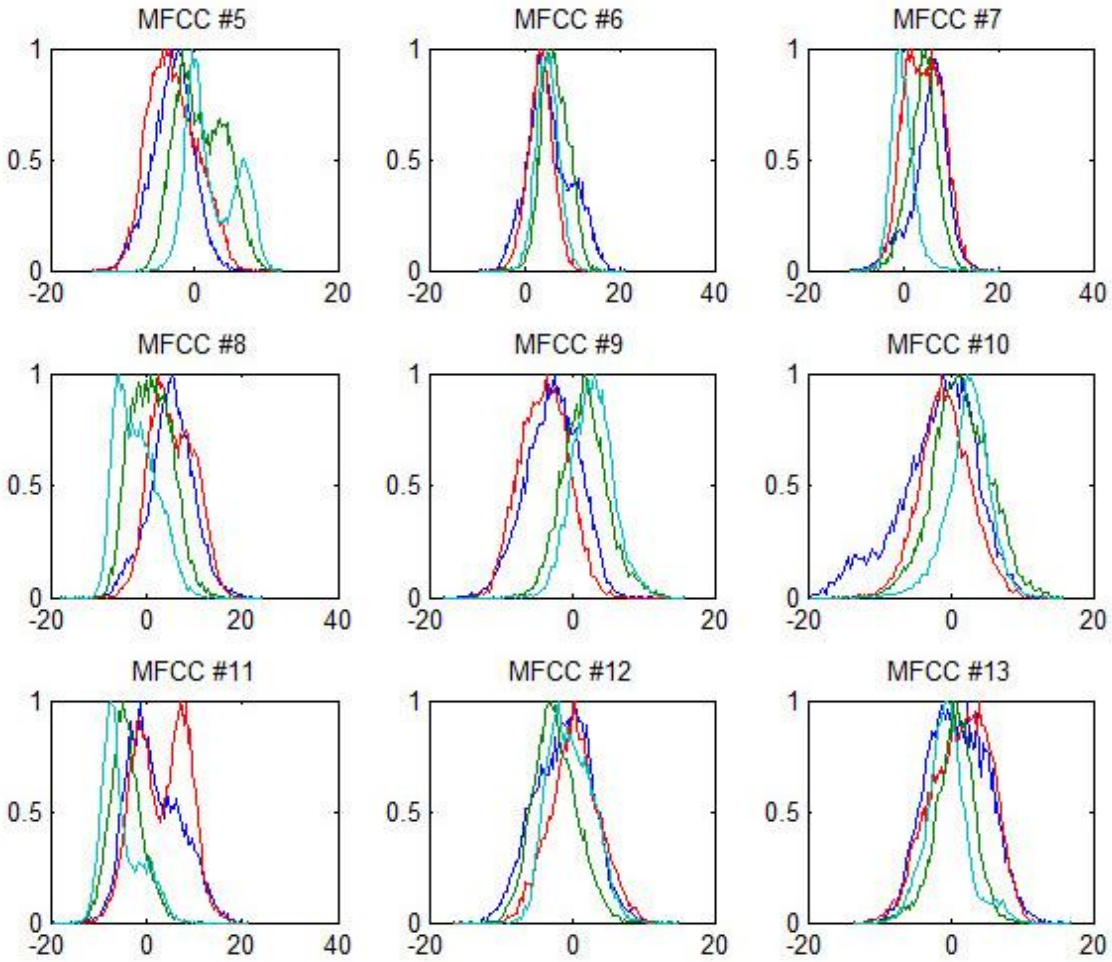


Figura 27. Histogrames dels MFCC

Aquestes característiques s'identifiquen en el programari desenvolupat com `ASBS_MFCC2` a `ASBS_MFCC13`. El seu càlcul s'implementa en la funció `asbsmfcc`.

3.3. GENERACIÓ DEL REPOSITORI

Per a la generació del repositori, es llegeix l'arxiu breathing.wav, es descompon en segments segons la longitud de finestra i, per a cada segment amb un valor de RMS major de 0.00125 (pàg. 27), es treuen totes les característiques descrites a l'apartat anterior i se l'assigna el moviment respiratori corresponent d'acord l'arxiu de marques temporals breathing.xlsx.

Aquest procés es realitza per a cada combinació de freqüència de mostreig i longitud de finestra, ja que els valors i nombre de característiques són diferents en funció d'aquests

paràmetres. El resultat de la generació del repositori és la creació dels següents arxius, que seran els repositoris sobre els quals es realitzarà la identificació de moviments respiratoris:

Taula 4. Noms dels repositoris en funció del mostreig i finestra

freq. de mostreig	longitud finestra	Nom model	Codificació
22.050 Hz	92 ms	breathing_c100s22w92.xlsx	s22w92
22.050 Hz	69 ms	breathing_c100s22w69.xlsx	s22w69
22.050 Hz	46 ms	breathing_c100s22w46.xlsx	s22w46
22.050 Hz	23 ms	breathing_c100s22w23.xlsx	s22w23
16.000 Hz	128 ms	breathing_c100s16w128.xlsx	s16w128
16.000 Hz	96 ms	breathing_c100s16w96.xlsx	s16w96
16.000 Hz	64 ms	breathing_c100s16w64.xlsx	s16w64
16.000 Hz	32 ms	breathing_c100s16w32.xlsx	s16w32
11.025 Hz	92 ms	breathing_c100s11w92.xlsx	s11w92
11.025 Hz	69 ms	breathing_c100s11w69.xlsx	s11w69
11.025 Hz	46 ms	breathing_c100s11w46.xlsx	s11w46
11.025 Hz	23 ms	breathing_c100s11w23.xlsx	s11w23
8.000 Hz	128 ms	breathing_c100s8w128.xlsx	s8w128
8.000 Hz	96 ms	breathing_c100s8w96.xlsx	s8w96
8.000 Hz	64 ms	breathing_c100s8w64.xlsx	s8w64
8.000 Hz	32 ms	breathing_c100s8w32.xlsx	s8w32

La generació dels repositoris s'implementa en les funcions `asbsgeneratemodel` i `asbsgenerateallmodels`.

Cada un dels arxius generats consta de tantes files com segments corresponents a tots els moviments respiratoris (fragmentats segons l'amplada de finestra) i 19 columnes, identificades com:

Taula 5. Identificació de les columnes del repositori

Columna	Contingut
1	ASBS_RMS
2	ASBS_CENTROID
3	ASBS_FLATNESS
4	ASBS_ROLLOFF
5	ASBS_ZCR
6	ASBS_MFCC2
7	ASBS_MFCC3
8	ASBS_MFCC4
9	ASBS_MFCC5
10	ASBS_MFCC6
11	ASBS_MFCC7
12	ASBS_MCFF8
13	ASBS_MFCC9
14	ASBS_MFCC10
15	ASBS_MFCC11
16	ASBS_MFCC12
17	ASBS_MFCC13
18	ASBS_RATIORMS
19	Moviment respiratori

Les primeres 18 corresponen als valors de cada una de les característiques descrites a l'apartat anterior. La columna 19 correspon a la descripció de el moviment respiratori, codificat de la següent manera:

Taula 6. Codi numèric assignat als moviments respiratoris

Codi	Descripció
1	Inspiració bucal
2	Espiració bucal
3	Inspiració nasal
4	Espiració nasal

Així doncs, els 1.784 moviments respiratoris continguts a l'arxiu base breathing.wav es converteixen en un nombre de trames, determinat en funció de la freqüència de mostreig i la longitud de la finestra. Aquest nombre de trames està comprès entre les 34.945 (freqüència de mostreig de 8.000 Hz i finestra de 128 ms) i les 191.220 (freqüència de mostreig de 22.050 Hz i finestra de 23 ms).

CAPÍTOL 4 - SISTEMA D'IDENTIFICACIÓ

4.1. INTRODUCCIÓ

L'objectiu principal d'aquest treball és estudiar i analitzar elements que són necessaris per a l'elaboració d'un sistema de detecció dels moviments respiratoris (inspiració i espiració, bucal i nasal) i implementar una solució per a detectar, en temps real, aquests moviments respiratoris a través d'un micròfon situat a prop de l'usuari.

El capítol 3 ha conclòs amb la creació d´un repositori de mostres identificades amb les seves característiques associades. Aquest capítol tracta sobre el desenvolupament del sistema d'identificació, és a dir, a partir de mostres desconegudes qualificades per les característiques temporals i freqüencials descrites en el capítol anterior, el sistema ha de predir el moviment respiratori que li correspon, fent servir com a base, el repositori.

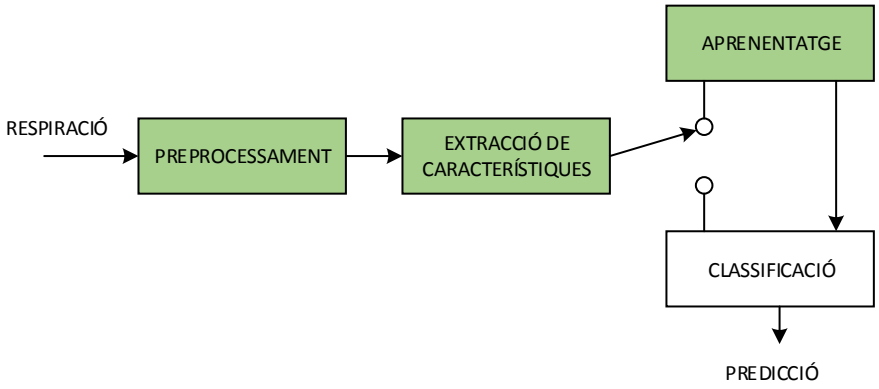


Figura 28. Procés d'aprenentatge (capítol 3)

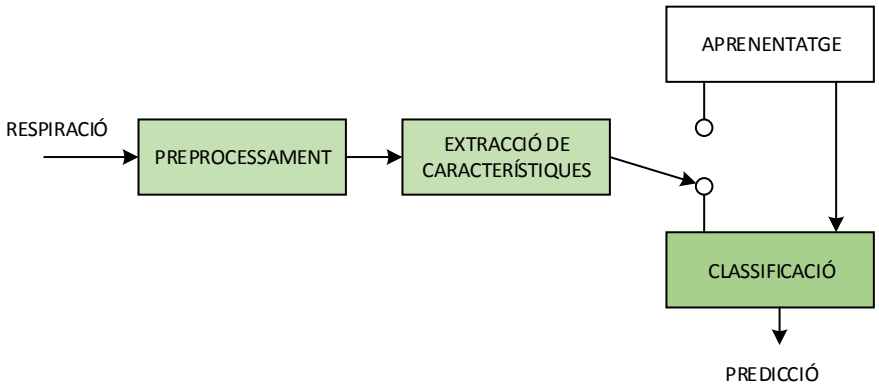


Figura 29. Procés d'identificació

A diferència d'altres tipus de problemes d'identificació en els que no hi ha relació entre les mostres desconegudes, la detecció dels moviments respiratoris parteix de què aquestes estan seqüenciades en el temps i, per tant, les mostres que s'envien al sistema mantenen aquest ordre. Això permet aprofitar aquesta propietat per a millorar el sistema d'identificació.

Així doncs, el sistema de detecció respiratori, estarà format per la combinació de dos components:

- Un component a l'espai, basat en la identificació del sistema davant mostres independents.
- Un component temporal, basat en la combinació de la resposta del sistema d'identificació de les mostres unitàries amb relació amb les respostes anteriors.

L'apartat 4.2. correspon al sistema d'identificació de mostres aïllades. Posteriorment, l'apartat 4.5. incrementarà la robustesa del sistema al afegir la història d'identificacions unitàries de les trames anteriors.

4.2. SELECCIÓ DEL SISTEMA DE CLASSIFICACIÓ

El sistema de decisió basat en trames independent se l'anomena sistema de classificació. L'elecció del sistema de classificació depèn de l'objectiu i les dades de què es disposen. En el cas d'aquest Treball, un sistema de decisió supervisat (aquell on s'ensenya al sistema el que ha de classificar) és el més idoni donat què es parteix de mostres identificades i l'objectiu és classificar noves mostres desconegudes.

Dintre dels sistemes supervisats, el mètode kNN [**16**] (k-veí-més-proper) és l'escollit per la seva facilitat d'aprenentatge, la flexibilitat d'ús davant el canvi del nombre i característiques a fer servir i l'alta precisió d'identificació davant altres mètodes en múltiples problemes d'identificació i classificació.

El mètode k-NN mesura la distància entre un ítem desconegut i un conjunt d'ítems coneguts que formen el model. Tant l'ítem desconegut com els ítems del model estan formats pel mateix tipus de característiques.

El mètode k-NN es basa en escollir els k ítems del model en què la distància amb l'ítem desconegut sigui mínima. Com l'ítem del model està identificat (etiquetat), aquesta etiqueta correspondrà a la identificació per l'ítem desconegut. En aquest projecte, k és 1.

Existeixen múltiples formes de calcular la mesura de la distància. La més habitual i la que es fa servir en aquest projecte és la distància Euclidiana.

La seva expressió és:

$$d(x,y) = \sqrt{\sum_{j=1}^{N}(x_j - y_j)^2} \tag{22}$$

On x_i i y_i són la característica j dels ítems x i y. N és el nombre total de característiques.

4.3. IMPLEMENTACIÓ DEL SISTEMA DE CLASSIFICACIÓ

La implementació del mètode kNN (1-NN en aquest cas), fa servir directament la funció `knnsearch` de Matlab:

```
I = knnsearch( MODEL(:, IDX), QUERY(:, IDX) )
```

On QUERY i MODEL són el model i els segments de respiració a classificar respectivament. IDX s'encarrega de seleccionar les característiques, entre les 18 possibles, que es tenen en compte en el procés.

D'aquesta manera, es permet molta flexibilitat a l'estudi de la robustesa de les cada una característica per separat o en qualsevol combinació.

La funció implementada a més alt nivell i que fa servir `knnsearch` per a la identificació dels segments de respiració és `asbslabconsole` i per a definir les característiques que es volen implicar en el procés, es passen com un únic paràmetre en forma de suma com, per exemple, `asbslabconsole(..., ASBS_CENTROID + ASBS_FLATNESS )`.

Un dels desavantatges que té el mètode kNN és que és d'un elevat cost de computació. Sense cap tipus d'optimització, l'algorisme ha de calcular la distància de la mostra desconeguda amb totes les mostres del model.

L'ús directe de la funció `knnsearch` no permetria disposar del sistema per a ser executar en temps real per a la majoria de combinacions de freqüència de mostreig i longitud de finestra. El temps de computació requerit per aquesta funció és molt alt (per exemple, 0.06 segons amb les 18 característiques i un model de 48.000 mostres) i el temps total del procés podria superar el temps de finestra (0.092 segons en el cas de l'exemple, que tenint en compte l'encavalcament, passen a set 0.046 segons).

En aquesta situació, hi hauria un retard cada vegada major entre que arriba l'àudio i es fa la identificació. Un retard que s'anés acumulant faria que la identificació del sistema no es pogués fer en temps real.

Per a resoldre això, Matlab disposa de la funció `createns`, per a la creació d'un objecte model de dades, que Matlab estructura internament per a optimitzar el càlcul de la distància. En la crida a la funció `knnsearch,` es referencia a aquest objecte i el temps de computació es redueix en un ordre de magnitud.

4.3.1. estandardització

Un inconvenient que pot presentar el càlcul de la distància dintre del mètode kNN és que els valors de les característiques estiguin dintre de rangs de diferents ordres de magnituds. Per exemple, s'ha vist com el marge obtingut dels valors dels coeficients MFCC (pàg. 35) oscil·la entre els -20 i +20 aproximadament, mentre que els valors de la resta de característiques tenen valors entre de magnitud molt més petita. Això provocaria, que en el càlcul de la distància Euclidiana, les característiques dels MFCC predominessin i deixessin la resta de característiques gairebé sense valor.

Una solució per a evitar-ho és l'estandardització dels valors de les característiques per tal de què els valors de totes elles estiguin compresos dins del mateix marge.

Per a estandarditzar els valors se substitueix cada valor de cada característica pel seu valor estandarditzat segons la següent expressió:

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{23}$$

On z_{ij} és el valor estandarditzat de la mostra i per a la característica j, x_{ij} el valor abans de

l'estandardització de la mateixa mostra, μ_j la mitjana de la característica j de totes les mostres i σ_j la desviació típica de la característica j de totes les mostres.

En entorns on es coneix per endavant els marges de les característiques per a totes les possibles mostres, l'estandardització és un mètode eficaç. També és eficaç en la validació de les mostres del model.

Ara bé, quan no es coneix el marge de possibles valors que pot prendre una característica determinada, l'estandardització no sempre dona els millors resultats. Amb tot, atenent a les proves realitzades, es preferible aplicar l'estandardització per tal d'evitar, en aquest cas, que els coeficients MFCC predominin en qualsevol càlcul de distància de l'algorisme kNN on intervinguin.

Així doncs, l'estandardització, s'inclou dintre de la funció `asbsloadmodel` i s'obtenen els valors de μ i σ per ser aplicats en les futures mostres desconegudes.

4.4. VALIDACIÓ DEL REPOSITORI

4.4.1. validació creuada

Per a validar el repositori, es fa servir la tècnica de la validació creuada. Aquí, es divideix el repositori en tres parts iguals. Un terç correspondrà a les mostres a validar i els altres dos terços correspondran a les dades del model. Aquesta verificació es realitza tres vegades, modificant els rols que juguen cadascuna de les particions.

El repositori d'aquest Treball, posseeix la particularitat de què les mostres (tot i que són independents unes d'altres) estan emmagatzemades d'acord l'ordre temporal en què van ser enregistrades i una partició en terços consecutiva no seria representativa, ja que possiblement un d'aquest contindria tots les mostres generades de la gravació d'un voluntari en concret.

Per a obtenir particions amb mostres no correlaciones, s'escullen els terços aleatòriament del conjunt de mostres del repositori.

La implementació de la validació la realitza la funció `asbsknnvalidate` i el resultat de la validació creuada és el següent:

Taula 7. Resultats de la validació creuada

Característica	Tot encertat (%)	Només bucal o nasal (%)	Només insp. o esp. (%)
RMS	25.83 ± 0.44	50.35 ± 0.44	51.34 ± 0.56
CENTROID	47.40 ± 0.27	84.35 ± 0.32	54.91 ± 0.09
FLATNESS	47.68 ± 0.46	84.75 ± 0.39	54.03 ± 0.57
ROLLOFF	44.24 ± 0.27	73.44 ± 0.66	57.66 ± 0.64
ZCR	42.13 ± 1.11	76.99 ± 0.89	53.95 ± 0.90
MFCC2	46.69 ± 0.37	86.64 ± 0.76	52.91 ± 0.40
MFCC3	33.85 ± 0.10	52.57 ± 0.31	57.92 ± 0.22
MFCC4	43.60 ± 0.38	71.00 ± 0.37	58.34 ± 0.21
MFCC5	36.76 ± 0.59	67.05 ± 0.38	53.08 ± 0.62
MFCC6	31.44 ± 0.34	54.96 ± 0.21	56.59 ± 0.55
MFCC7	35.62 ± 0.38	57.62 ± 0.65	58.49 ± 0.39
MFCC8	35.72 ± 0.28	65.22 ± 0.52	52.69 ± 0.17
MFCC9	39.08 ± 0.34	72.98 ± 0.38	52.30 ± 0.60
MFCC10	31.53 ± 0.49	56.88 ± 0.08	53.40 ± 0.21
MFCC11	38.42 ± 0.63	69.76 ± 0.65	53.46 ± 0.75
MFCC12	28.47 ± 0.48	51.82 ± 0.58	54.29 ± 0.24
MFCC13	28.60 ± 0.54	54.57 ± 0.16	50.71 ± 0.56
RMSRATIO	48.64 ± 0.26	87.53 ± 0.20	55.56 ± 0.43
Tots els MFCC	95.87 ± 0.35	98.73 ± 0.03	96.59 ± 0.36
Tot	96.64 ± 0.22	98.81 ± 0.12	97.35 ± 0.19

De la taula es desprèn que les característiques utilitzades de forma independent donen pocs resultats per sí soles. Els percentatges per a la identificació amb la utilització d'una única característica no són extremadament bons si es té en compte que d'acord a les probabilitats, els encerts tendirien a ser 25%, 50% i 50%, ja que es tracta de 4 solucions possibles en el primer cas i dues solucions possibles en els altres dos casos.

Tot i això, alguns d'ells arriben quasi al 50% d'encerts en la primera columna, duplicant el percentatge propi de l'atzar. Destaca que el que té millor percentatge de precisió és el RMSRATIO, que correspon a la característica que experimentalment es va definir i analitzar per a fixar una freqüència sobre la qual calcular l'energia acumulada (pàg. 34).

Per altra banda, si es fa servir la combinació de característiques, el percentatge d'encert s'incrementa de forma espectacular. A la taula, s'ha inclòs els resultats quan es combinen tots els MFCC per un costat i totes les 18 característiques per l'altre. Els percentatges superen el 96 %.

Els resultats anteriors, es basen en la versió del model amb una freqüència de mostreig de 22.050 Hz i longitud de finestra de 92 ms. Agafant un terç de les mostres i tornant a fer la validació per a la resta de combinacions freqüència de mostreig - longitud de finestra, el resultat és el següent:

Taula 8. Percentatges d'identificació per a la versió en 22.050 Hz del model

Característica	22.050 Hz 92 ms	22.050 Hz 69 ms	22.050 Hz 46 ms	22.050 Hz 23 ms
RMS	25.83 %	26.14 %	26.33 %	25.83 %
CENTROID	47.40 %	47.18 %	46.83 %	46.06 %
FLATNESS	47.68 %	47.55 %	47.86 %	46.64 %
ROLLOFF	44.24 %	42.66 %	40.22 %	34.18 %
ZCR	42.13 %	38.47 %	35.94 %	33.88 %
MFCC2	46.69 %	47.26 %	47.11 %	46.21 %
MFCC3	33.85 %	34.21 %	33.67 %	32.99 %
MFCC4	43.60 %	43.56 %	42.75 %	41.24 %
MFCC5	36.76 %	35.89 %	35.67 %	33.85 %
MFCC6	31.44 %	30.86 %	30.07 %	29.06 %
MFCC7	35.62 %	35.15 %	34.17 %	32.49 %
MFCC8	35.72 %	35.86 %	35.29 %	33.91 %
MFCC9	39.08 %	38.16 %	37.25 %	35.19 %
MFCC10	31.53 %	30.81 %	30.22 %	29.31 %
MFCC11	38.42 %	37.79 %	37.39 %	35.07 %

MFCC12	28.47 %	27.84 %	27.44 %	26.67 %
MFCC13	28.60 %	28.74 %	27.52 %	27.18 %
RMSRATIO	48.64 %	48.94 %	48.31 %	47.64 %
Tots els MFCC	95.87 %	95.81 %	93.86 %	93.99 %
Tot	96.64 %	96.51 %	95.41 %	91.86 %

Taula 9. Percentatges d'identificació per a la versió en 16.000 Hz del model

Característica	16.000 Hz 128 ms	16.000 Hz 96 ms	16.000 Hz 64 ms	16.000 Hz 32 ms
RMS	25.68 %	25.72 %	26.16 %	26.23 %
CENTROID	44.99 %	45.17 %	45.06 %	44.02 %
FLATNESS	45.29 %	46.21 %	46.76 %	45.44 %
ROLLOFF	41.11 %	39.14 %	35.80 %	32.25 %
ZCR	41.01 %	40.11 %	38.26 %	32.58 %
MFCC2	45.01 %	46.33 %	45.92 %	45.32 %
MFCC3	36.92 %	38.29 %	37.71 %	37.07 %
MFCC4	45.89 %	45.69 %	45.17 %	41.91 %
MFCC5	31.25 %	31.61 %	31.51 %	30.05 %
MFCC6	31.44 %	31.78 %	30.97 %	29.98 %
MFCC7	38.57 %	38.95 %	38.77 %	37.22 %
MFCC8	37.37 %	36.81 %	36.22 %	34.20 %
MFCC9	30.61 %	30.56 %	30.16 %	28.88 %
MFCC10	36.69 %	37.29 %	36.91 %	35.26 %
MFCC11	26.96 %	26.85 %	26.84 %	26.54 %
MFCC12	33.87 %	33.55 %	32.27 %	30.46 %
MFCC13	26.09 %	26.10 %	26.53 %	25.84 %
RMSRATIO	48.32 %	48.00 %	48.46 %	47.55 %
Tots els MFCC	95.77 %	95.67 %	94.61 %	90.06 %
Tot	96.87 %	96.66 %	95.98 %	92.79 %

Taula 10. Percentatges d'identificació per a la versió en 11.025 Hz del model

Característica	11.025 Hz 92 ms	11.025 Hz 69 ms	11.025 Hz 46 ms	11.025 Hz 23 ms
RMS	25.73 %	26.22 %	26.50 %	26.11 %
CENTROID	45.49 %	44.79 %	44.46 %	43.82 %
FLATNESS	47.69 %	47.38 %	46.61 %	45.64 %
ROLLOFF	39.42 %	38.26 %	30.80 %	23.93 %
ZCR	37.05 %	35.11 %	32.88 %	30.95 %
MFCC2	46.76 %	47.07 %	45.80 %	45.00 %
MFCC3	45.04 %	44.68 %	44.91 %	42.54 %
MFCC4	36.47 %	36.09 %	35.03 %	33.25 %
MFCC5	30.61 %	30.79 %	29.80 %	28.44 %
MFCC6	46.53 %	46.78 %	45.11 %	41.98 %
MFCC7	38.99 %	38.63 %	37.04 %	34.45 %
MFCC8	29.06 %	28.21 %	27.97 %	27.14 %
MFCC9	34.06 %	33.67 %	33.42 %	32.03 %
MFCC10	29.38 %	28.88 %	28.20 %	26.88 %
MFCC11	27.89 %	27.98 %	28.97 %	27.18 %
MFCC12	27.51 %	27.07 %	26.30 %	26.04 %
MFCC13	31.77 %	31.56 %	30.64 %	28.44 %
RMSRATIO	48.79 %	49.08 %	48.25 %	47.39 %
Tots els MFCC	95.14 %	93.91 %	91.26 %	83.65 %
Tot	96.51 %	95.77 %	93.97 %	87.89 %

Taula 11. Percentatges d'identificació per a la versió en 8.000 Hz del model

Característica	8.000 Hz 128 ms	8.000 Hz 96 ms	8.000 Hz 64 ms	8.000 Hz 32 ms
RMS	26.45 %	26.51 %	26.75 %	26.15 %
CENTROID	36.89 %	37.56 %	37.04 %	35.65 %
FLATNESS	42.44 %	42.62 %	41.77 %	41.07 %

ROLLOFF	36.46 %	33.37 %	27.23 %	21.37 %
ZCR	31.10 %	29.97 %	29.83 %	28.23 %
MFCC2	38.60 %	38.49 %	38.55 %	37.23 %
MFCC3	47.71 %	47.52 %	47.02 %	44.78 %
MFCC4	35.16 %	33.84 %	33.93 %	32.20 %
MFCC5	43.52 %	44.32 %	43.02 %	40.37 %
MFCC6	29.03 %	29.59 %	29.12 %	28.08 %
MFCC7	28.99 %	28.74 %	28.59 %	27.82 %
MFCC8	37.52 %	35.69 %	36.17 %	34.44 %
MFCC9	30.01 %	30.17 %	30.08 %	28.06 %
MFCC10	27.64 %	27.80 %	28.15 %	27.26 %
MFCC11	33.86 %	33.33 %	31.21 %	30.27 %
MFCC12	28.06 %	27.85 %	28.11 %	26.84 %
MFCC13	28.00 %	28.01 %	27.34 %	26.57 %
RMSRATIO	45.70 %	45.82 %	45.55 %	43.88 %
Tots els MFCC	94.71 %	94.05 %	91.99 %	84.55 %
Tot	96.22 %	95.95 %	94.58 %	89.07 %

Del resultat de les taules es pot destacar el següent:

- Els millors resultats s'aconsegueixen amb les combinacions 22.050 Hz – 92 ms i 16.000 Hz – 128 ms. Entre aquestes dues combinacions, hi ha característiques que donen millor resultats en un cas i viceversa. En temps de computació, l'opció 16.000 Hz – 128 ms és més ràpida: el nombre de mostres per segon es redueix en 6.050, l'amplada de la finestra agafa 36 ms més i el nombre de mostres en el repositori es redueix (passa de 48.685 a 35.361), reduint el temps de computació de l'algorisme kNN.
- La reducció de la freqüència de mostreig (a partir dels 16.000 Hz) fa decréixer el percentatge d'encerts, però d'una manera molt poc pronunciada i gairebé no perceptible si es considera totes les característiques.

- La reducció de la longitud de la finestra afecta de manera més pronunciada al percentatge d'encerts. Considerant totes les característiques, para cada freqüència de mostreig, les reduccions van del 96.64 % a 91.86 % (per a 22.050 Hz), de 96.87 % a 92.79 % (per a 16.000 Hz), de 96.51 % a 87.89 % (per a 11.025 Hz) i de 96.22 % a 89.07 % (per a 8.000 Hz). De fet, en temps de computació és un avantatge, ja que la reducció de la longitud de finestra no només incrementa el temps de computació sinó que, a més, degrada el nivell de precisió.

Del treball previ d'anàlisi de les característiques (apartat 3.2.5.) podrien intuir-se els resultats exposats. Les gràfiques amb les distribucions dels valors de les característiques mostren que el so respiratori és més predominant a les baixes freqüències. Per tant, la reducció de la freqüència de mostreig no afecta.

De manera anàloga, la reducció en la longitud de la finestra sí que afecta en la identificació, ja que aquesta reducció, fa perdre informació estructural en les baixes freqüències.

4.4.2. Combinacions de característiques

Les taules de validació anteriors estan construïdes considerant l'ús de les característiques de manera independent i dues combinacions possibles: tots els MFCC junts per una banda i totes les característiques per l'altre.

Però, amb 18 característiques, les possibles combinacions que es poden escollir són moltes més, en concret, 262.143 combinacions, calculades de l'expressió:

$$NComb = \sum_{n=1}^{18} \binom{18}{n} = 262.143 \tag{24}$$

Si es volgués escollir la millor combinació amb n característiques simultànies, la selecció, des de les taules anteriors, de les n característiques amb els millors percentatges individuals d'encerts no és una garantia. Poden haver-hi combinacions amb característiques menys independentment robustes però que, en certes combinacions, donin millor resultats. Per això, s'ha cregut interessant veure el percentatge d'encerts per a qualsevol combinació de característiques. Ja que això comporta un cost computacional mesurat en dies, s'ha utilitzat

un altre ordinador que ha realitzat aquests càlculs de forma paral·lela al desenvolupament del treball.

Aquesta avaluació, s'implementa en la funció `asbsknnvalidateall`, que genera totes les combinacions possibles, calcula el percentatge d'identificacions positives per a cada una d'elles i emmagatzema tota aquesta informació a l'arxiu Excel results2292.xlsx. S'ha parametritzat el procés d'identificació amb una freqüència de mostreig de 22.050 Hz i una longitud de finestra de 92 ms.

Els resultats es poden veure resumits en la següent figura:

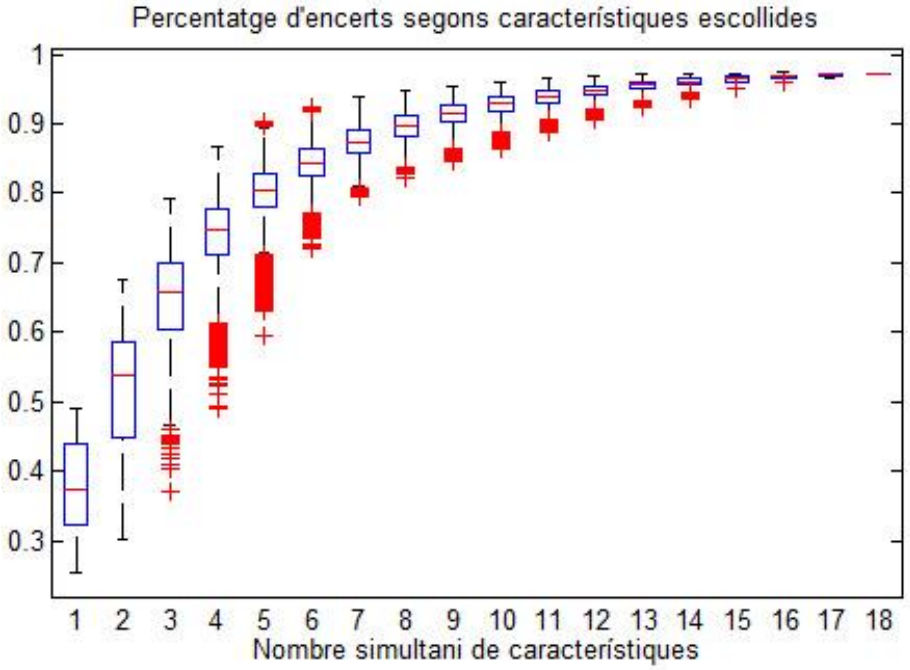


Figura 30. Encerts segons característiques i el seu nombre.

A la figura es veu clarament que a mesura que s'incrementa el nombre simultani de característiques, la mitjana del percentatge d'encerts augmenta considerablement. Però també es desprèn que, per exemple, 4 característiques ben seleccionades donen millor resultats que 10 mal escollides, amb el corresponent estalvi de cost computacional que suposaria.

Per altra banda, quan les possibles combinacions són moltes, el marge del percentatge d'encerts entre la millor i pitjor combinació és ampli.

De forma més detallada, les combinacions més i menys exitoses, juntament amb els seus percentatges d'encert, es mostra en la següent taula:

Taula 12. Millors i pitjors combinacions de característiques

#		Encerts	RMS	SC	SF	RO	ZCR	MFCC2	MFCC3	MFCC4	MFCC5	MFCC6	MFCC7	MFCC8	MFCC9	MFCC10	MFCC11	MFCC12	MFCC13	RÀTIO
1	Min	25,53%	✓																	
	Max	49,03%																		✓
2	Min	30,18%	✓															✓		
	Max	67,54%		✓																✓
3	Min	37,08%	✓														✓	✓		
	Max	79,12%		✓				✓												✓
4	Min	48,97%	✓								✓						✓	✓		
	Max	86,57%				✓		✓	✓							✓				
5	Min	58,48%	✓								✓				✓		✓	✓		
	Max	90,23%	✓			✓		✓	✓							✓				
6	Min	71,96%	✓							✓	✓				✓		✓	✓		
	Max	92,43%	✓		✓			✓	✓		✓					✓				
7	Min	79,58%	✓							✓	✓			✓	✓		✓	✓		
	Max	93,84%	✓		✓			✓	✓		✓	✓				✓				
8	Min	82,29%		✓	✓	✓	✓	✓			✓				✓		✓			
	Max	94,72%	✓					✓	✓	✓	✓	✓				✓	✓			
9	Min	84,53%		✓	✓	✓	✓	✓			✓			✓			✓			✓
	Max	95,50%	✓		✓			✓	✓		✓	✓			✓	✓	✓			
10	Min	86,49%		✓	✓	✓	✓	✓			✓			✓	✓		✓			✓
	Max	96,12%	✓		✓			✓	✓	✓	✓	✓			✓	✓	✓			
11	Min	88,72%		✓	✓	✓	✓	✓			✓			✓	✓		✓			✓
	Max	96,58%	✓					✓	✓	✓	✓	✓			✓	✓	✓	✓		✓
12	Min	90.48%		✓	✓	✓	✓	✓			✓	✓			✓		✓	✓		✓
	Max	97,02%	✓			✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓
13	Min	92,27%		✓	✓	✓	✓	✓			✓	✓	✓		✓	✓		✓	✓	✓
	Max	97,11%	✓		✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
14	Min	93,74%	✓	✓	✓	✓	✓	✓			✓	✓	✓		✓	✓		✓	✓	✓
	Max	97,24%	✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
15	Min	94,96%		✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Max	97,30%	✓				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
16	Min	95,87%	✓	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Max	97,33%	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
17	Min	96,66%	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Max	97,29%	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
18		97,24%	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Es pot veure com, a partir de 5 característiques, totes les millors opcions inclouen l'RMS. En principi sembla contradictori que aquesta característica formi part de les millors combinacions quan, tal com es va dir a l'apartat 3.1.3. , aquesta mesura varia molt en funció de la sensibilitat del micròfon i la distància d'aquest al l'usuari. Però, el tipus de validació realitzat és una validació creuada, en què existeixen mostres del mateix voluntari en tots dos conjunts.

Sembla doncs, interessant, mostrar de nou la taula, però sense tenir en consideració la característica RMS:.

Taula 13. Millors i pitjors combinacions de característiques (sense RMS)

#		Encerts	SC	SF	RO	ZCR	MFCC2	MFCC3	MFCC4	MFCC5	MFCC6	MFCC7	MFCC8	MFCC9	MFCC10	MFCC11	MFCC12	MFCC13	RATIO
1	Min	28.32%														✓			
	Max	49,03%																	✓
2	Min	33.29%														✓	✓		
	Max	67,54%	✓																✓
3	Min	42.47%								✓						✓	✓		
	Max	79,12%	✓				✓												✓
4	Min	52.25%								✓				✓		✓	✓		
	Max	86,57%			✓		✓	✓							✓				
5	Min	64.17%							✓	✓				✓		✓	✓		
	Max	89.37%		✓			✓	✓		✓					✓				
6	Min	73.91%							✓	✓			✓	✓		✓	✓		
	Max	91.53%		✓			✓	✓	✓	✓					✓				
7	Min	79.65%	✓	✓		✓	✓			✓				✓		✓			
	Max	93.14%					✓	✓	✓	✓				✓	✓				
8	Min	82,29%	✓	✓	✓	✓	✓			✓				✓		✓			
	Max	94,24%						✓	✓		✓	✓		✓	✓	✓			✓
9	Min	84,53%	✓	✓	✓	✓	✓			✓			✓			✓			✓
	Max	95,23%						✓	✓	✓	✓			✓	✓	✓			✓
10	Min	86.49%	✓	✓	✓	✓	✓			✓			✓	✓		✓			✓
	Max	95,87%						✓	✓	✓	✓			✓	✓	✓	✓		✓
11	Min	86.72%	✓	✓	✓	✓	✓	✓		✓			✓	✓		✓			✓
	Max	96,32%						✓	✓	✓	✓	✓		✓	✓	✓	✓		✓
12	Min	90,48%	✓	✓	✓	✓	✓	✓		✓	✓			✓		✓	✓		✓
	Max	96.79%						✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
13	Min	92.27%	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓		✓	✓		✓
	Max	96.77%					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓
14	Min	93.87%	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓		✓	✓	✓
	Max	96.89%				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
15	Min	94.96%	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Max	96,94%		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
16	Min	95,97%	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Max	96,96%		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
17		96,95%	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

La taula, mostra com, a mesura que s'incrementa el nombre de característiques simultànies, els coeficients MFCCs formen part de les combinacions amb més percentatge d'encerts.

Els resultats de la taula de dalt, no signifiquen que les combinacions mostrades com més òptimes siguin les mateixes per a totes les situacions i escenaris, però sí que poden servir com orientatives.

4.4.3. Validació amb mostres desconegudes

El conjunt de validacions del model fet enguany, s'han basat en l'encreuament de mostres del repositori. El pas següent, és provar l'efectivitat del model per a mostres desconegudes.

Per a fer la validació, s'han tret totes les mostres d'un voluntari, que s'han considerat com a mostres desconegudes i s'ha procedit a validar-les en el sistema. El resultat ha sigut el següent:

Taula 14. Resultats de validació amb mostres desconegudes

Característica	Tot encertat (%)	Només bucal o nasal (%)	Només insp. o esp. (%)
RMS	26.11 %	50.13 %	51.76 %
CENTROID	45.62 %	87.04 %	51.96 %
FLATNESS	44.55 %	87.46 %	50.33 %
ROLLOFF	34.45 %	65.68 %	52.24 %
ZCR	31.79 %	73.30 %	45.34 %
MFCC2	45.89 %	89.31 %	50.22 %
MFCC3	21.05 %	44.17 %	49.74 %
MFCC4	31.90 %	66.50 %	47.36 %
MFCC5	27.74 %	57.37 %	50.44 %
MFCC6	28.36 %	57.92 %	51.03 %
MFCC7	36.19 %	65.38 %	54.62 %
MFCC8	28.18 %	58.40 %	48.48 %
MFCC9	36.63 %	75.65 %	48.09 %
MFCC10	24.57 %	47.32 %	52.38 %
MFCC11	38.19 %	74.18 %	50.55 %
MFCC12	25.25 %	50.26 %	52.46 %
MFCC13	18.50 %	37.26 %	50.02 %
RMSRATIO	44.37 %	90.23 %	49.30 %
Tots els MFCC	69.53 %	95.67 %	72.22 %
Tot	64.65 %	96.11 %	67.16 %

Del resultat es desprèn que els percentatges d'identificació positiva del sistema no són tan alts com abans. El 69.53 % d'encerts al qual s'arriba amb l'ús de tots el coeficients MFCC és una bona fita, però tractant-se de 4 possibles resultats, el sistema haurà de ser capaç d'incrementar aquest percentatge.

Destaca, per altra banda, l'alt índex d'encerts alhora de discernir entre respiració bucal o nasal, resultat que semblava previsible segons tal com es va veure gràficament a les distribucions de l'apartat 3.2.5.).

La següent taula, mostra la matriu de confusió quan es fan servir totes les característiques en la identificació:

Taula 15. Matriu de confusió per a mostres desconegudes

	Insp. bucal	**Esp. bucal**	**Insp. nasal**	**Esp. nasal**	**Total**
Insp. bucal	**1.015**	207	8	25	**1.255**
Esp. bucal	1.522	**1.508**	20	120	**3.170**
Insp. nasal	105	72	**1.857**	261	**2.295**
Esp. nasal	4	40	1.500	**2.200**	**3.744**
Total	**2.646**	**1.827**	**3.385**	**2.606**	**10.464**

La matriu de confusió mostra com on radica un repte important en la identificació positiva és en la diferenciació entre inspiració i espiració dins d'un mateix tipus de respiració (bucal o nasal). Per altra banda, la identificació de si la respiració és nasal o bucal és molt robusta, amb un percentatge d'encerts molt alt (més del 96 %).

Si es fan servir les combinacions millors i pitjors mostrades en la Taula 13, es pot comprovar, per una banda, fins a quin punt les combinacions de la taula poden prendre's com orientatives i, per l'altra banda, una idea del nivell d'identificacions positives en funció del nombre de característiques i sense haver de provar totes les combinacions possibles.

Els resultats d'aplicar la identificació amb les combinacions de característiques de la taula són els següents:

Taula 16. Millors i pitjors combinacions de característiques amb mostres desconegudes

Nombre de característiques	Encerts (%) Pitjor combinació	Encerts (%) Millor combinació
1	25.25 %	44.37 %
2	20.87 %	54.22 %
3	25.41 %	61.28 %
4	23.63 %	63.02 %
5	29.78 %	65.57 %
6	44.08 %	69.31 %
7	43,40 %	69.64 %
8	45,67 %	66.98 %
9	51.83 %	69.13 %
10	57.74 %	65,28 %
11	57.59 %	62.26 %
12	53.19 %	69.78 %
13	65.18 %	69.60 %
14	66.04 %	69.45 %
15	59.08 %	69.31 %
16	61.68 %	69.44 %
17	69.03 %	

Els resultats de la taula mostren coherència en l'elecció de les combinacions òptimes i pèssimes en relació la Taula 13. La taula ha servit per a mostrar que amb només 6 característiques ben escollides ja s'arriba a un percentatge d'encerts quasi semblant que amb les 17 característiques. Potser hi ha una combinació millor, però sense provar tots els casos possibles s'arriba a un nivell semblant a l'aconseguit amb 17 característiques.

4.5. SEQÜÈNCIA TEMPORAL

Enguany, tot el sistema d'identificació està basat en la classificació de les mostres tractades de forma independent unes d'altres, mostres que corresponen a segments, de molt curta duració, de la respiració. Si al procés d'identificació estudiat i implementat fins ara mitjançant el mètode kNN s'afegeix un procés que tingui en compte la seqüencia temporal dels moviments respiratoris, el nivell d'identificació augmentarà.

La idea es basa en el fet que en un supòsit en el qual el procés de classificació amb el mètode kNN tingués un percentatge d'encerts del 100%, el resultat, en el temps, d'una seqüencia desconeguda de moviments respiratoris correspondria a una seqüència sense canvis sobtats de 0s, 1s, 2s 3s i 4s, cada un d'ells identificant el tipus corresponent de moviment respiratori.

Per exemple, per una freqüència respiratòria de 20 respiracions per minut (40 moviments respiratoris per minut) i una longitud de finestra de 92 ms, el nombre de segments per moviment respiratori és aproximadament de 20. Llavors, la identificació ideal de la seqüència "inspiració bucal - espiració bucal - inspiració nasal - espiració nasal" podria ser tal com:

00111111111111100000222222222200003333333333300044444444440000

(Els 0s pertanyen a les transicions, on el nivell d'energia està per sota del llindar i el sistema ho considera com a transició).

Ara bé, en realitat, s'ha vist que el sistema d'identificació basat en les mostres de forma independent no té un 100% d'encerts. Llavors, en el cas de l'exemple, el resultat de la identificació podria ser quelcom semblant a:

00122112241111110000022111222122000031133334443300043334144444000 0

Ara, amb aquest resultat, el percentatge d'encerts a l'escala de mostres, es quedaria en el 62%, però l'objectiu és identificar el moviment respiratori en el seu conjunt de mostres. Si es té en compte, entre altres indicadors, el resultat predominant entre transicions. Els valors predominants són 1,2,3,4. i, amb aquests valors, els moviments respiratoris han quedat perfectament identificats.

Aquest és un exemple senzill de com es pot millorar la precisió d'encert d'un moviment respiratori. En una situació real pot passar, per exemple, que hi hagi més d'un resultat

predominant entre dues transicions, que no hi hagi el 0 que indica transició o altres circumstàncies.

Però, en l'anàlisi de la seqüència de resultats, es pot mesurar no només el nombre predominant de valors entre transicions sinó altres indicadors. Per exemple, l'energia que es calcula per a cada segment sol tenir un decaïment més lent en les espiracions que en les inspiracions:

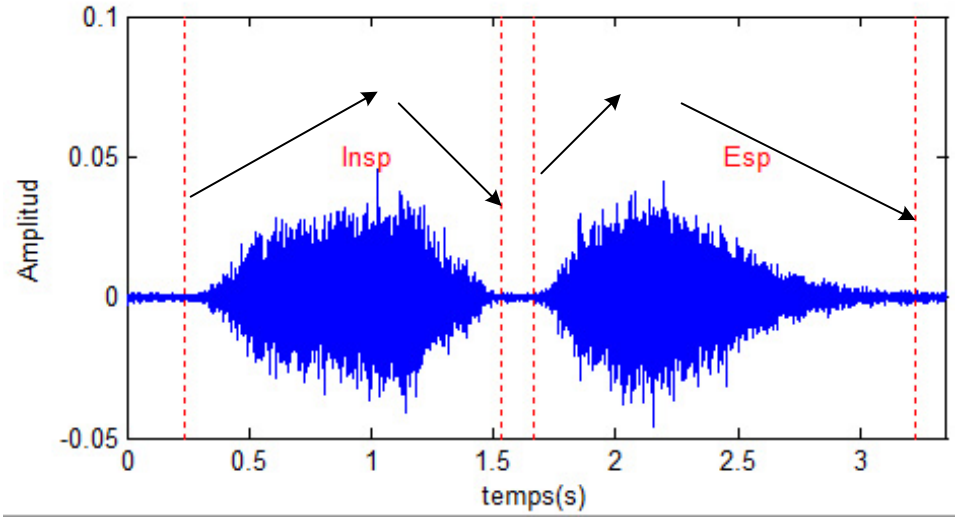


Figura 31. Fases de creixement i decaïment

L'energia i la resta d'indicadors que es faran servir per a identificar el moviment respiratori a partir de la seqüència de resultats procedents de la identificació de les mostres pel mètode kNN són:

- El nombre consecutiu de segments entre dos '0's amb energia decreixent.
- El nombre consecutiu de segments entre dos '0's amb energia creixent.
- El nombre d'1's entre dos '0's.
- El nombre de '2's entre dos '0's.
- El nombre de '3's entre dos '0's.
- El nombre de '4's entre dos '0's.
- El nombre consecutius de valors que no són '0'.
- El temps transcorregut des de l'últim '0' fins al moment actual.

El mecanisme de processament de la seqüència de resultats d'identificació de mostres per a arribar a la identificació de el moviment respiratori, es realitza mitjançant una màquina d'estats, el diagrama de la qual és el següent:

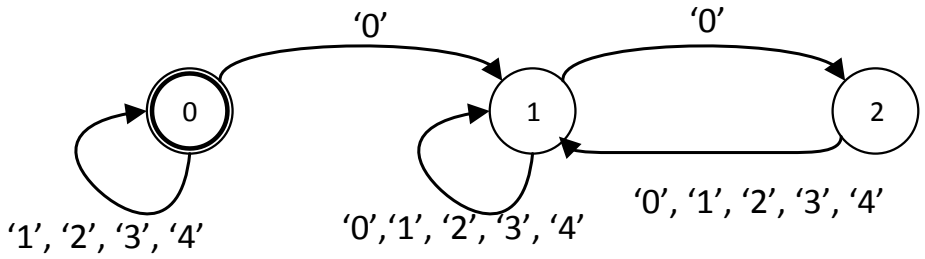


Figura 32. Diagrama d'estats

L'estat 0 és l'estat preliminar, un estat d'espera fins a l'inici d'un moviment respiratori. Posa els indicadors a zero i quan arriba un '0', moment el qual considera que comença un moviment respiratori, passa a estat 1.

L'estat 1 és el que comptabilitza la seqüència de valors corresponents des de l'inici del moviment respiratori fins el final. Actualitza els indicadors de la màquina d'estat d'acord als valors d'entrada i realitza les següent accions en funció dels indicadors:

- Si arriba un '0' després de què hagués arribat un altra '0' significa que encara està en fase de transició i no de finalització de moviment respiratori.
- Si arriba un '0' i el nombre de valors diferents de '0' comptabilitzat és menor de que un valor determinat, es considera informació insuficient o un pic de soroll i, per tant no ho considera moviment respiratori i passa a estat 2.
- Si arriba un '0' i el nombre de valors diferents de '0' comptabilitzat és major o igual que un valor fixat, determina el tipus de moviment respiratori en funció de següent algorisme:

```
total = nombre_'1's + nombre_'2's + nombre_'3's + nombre_'4's
p1 = nombre_'1's / N
p2 = nombre_'2's / N
p3 = nombre_'3's / N
p4 = nombre_'4's / N
pBucal = (nombre_'1's + nombre_'2's) / total
pNasal = (nombre_'3's + nombre_'4's) / total
pIns = (nombre_'1's + nombre_'3's) / total
pEsp = (nombre_'2's + nombre_'4's) / total
```

```
total2 = nombre_energia_seguit_creixent +
                    nombre_energia_seguit_decreixent
pIns2 =  nombre_energia_seguit_creient / total2
pEsp2 =  nombre_energia_seguit_decreient / total2
pBucalIns = 0.70 * p1 + 0.10 * pBucal + 0.10 * pIns + 0.10 * pIns2;
pBucalEsp = 0.70 * p2 + 0.10 * pBucal + 0.10 * pEsp + 0.10 * pEsp2;
pNasalIns = 0.70 * p3 + 0.10 * pNasal + 0.10 * pIns + 0.10 * pIns2;
pNasalEsp = 0.70 * p4 + 0.10 * pNasal + 0.10 * pEsp + 0.10 * pEsp2;

result = max([pBucalIns pBucalEsp pNasalIns pNasalEsp]);
```

Com es pot veure, la resolució final de el moviment respiratori està en funció dels següents factors:

- El valor més predominant amb un pes del 70 %.
- La combinació bucal o nasal predominant amb un pes del 10 %.
- La combinació inspiració o espiració predominant amb un pes del 10 %.
- El creixement o decaïment amb un pes del 10 %.

Finalment, l'estat 2 indica que ja s'ha realitzat una identificació, s'inicien els indicadors i davant qualsevol nou valor, passa a estat 1.

4.6. VALIDACIÓ

Per a validar la idoneïtat del sistema es fan proves de validació amb un enregistrament que no forma part del repositori. La seqüència de moviments respiratoris de test (testunknown_c100s22.wav), de durada d'1 minut i 20 segons està format per 12 respiracions nasals seguits de 12 respiracions bucals.

La taula de resultats de la identificació és la següent:

Taula 17. Identificacions positives amb característiques aïllades

Característica	Encerts
RMS	29.17 %
CENTROID	58.33 %
FLATNESS	60.42 %

ROLLOFF	27.08 %
ZCR	16.67 %
MFCC2	58.33 %
MFCC3	12,50 %
MFCC4	10,42 %
MFCC5	27.08 %
MFCC6	14.58 %
MFCC7	50.00 %
MFCC8	43.75 %
MFCC9	54.17 %
MFCC10	25.00 %
MFCC11	31.25 %
MFCC12	12.50 %
MFCC13	8.33 %
RMSRATIO	56.25 %
Tots els MFCC	93.75 %
Tot	95.83 %

Els resultats no són gaire satisfactoris si es fan servir les característiques de forma aïllada, però, quan es fan servir amb simultaneïtat, el percentatge d'encerts és molt alt (més del 93%).

Si s'apliquen les combinacions de característiques descrites a la Taula 13, els resultats són els següents:

Taula 18. Identificacions positives amb característiques combinades

Nombre de característiques	Encerts Millor combinació
1	56.25 %
2	66.67 %
3	91.67 %

4	72.92 %
5	66.67 %
6	72.92 %
7	81.25 %
8	89.58 %
9	85,42 %
10	70,83 %
11	77.08 %
12	95.83 %
13	95.83 %
14	93.75 %
15	93.75 %
16	93.75 %
17	95.83 %

Es pot veure com la combinació de característiques permet uns índex d'identificació positives de moviments respiratoris molt alts. Destaca, en la prova realitzada, que amb només 3 característiques (Centroide, MFCC3 i RMSRatio) el percentatge d'encert arriba al 91.67%, després decreix i no es torna a superar fins quan s'utilitzen 12 característiques simultànies. Com es va dir a l'apartat 4.4.2. , les combinacions utilitzades no tenen per què ser les òptimes en totes les situacions. Per això, aquest Treball permet, en la solució desenvolupada, definir en qualsevol moment, el conjunt de característiques que es volen fer servir.

4.7. COST COMPUTACIONAL

Un dels objectius principals i reptes del projecte és que fos possible la identificació dels moviments respiratoris en temps real, és a dir, des d'un micròfon. Això vol dir, que el temps de processament no pot superar el ritme d'enregistrament. Per exemple, si l'enregistrament es parametritza amb una freqüència de mostreig de 22.050 Hz, implica que el sistema ha de processar 22.050 mostres (enfinestrat, encavalcament, filtratge, extracció de

característiques, classificació kNN i identificació final) en no més d'un segon de temps per a evitar la pèrdua d'informació d'àudio.

A més, el nombre de segments a tractar dependrà de la longitud de la finestra, el temps de computació de l'extracció de característiques dependrà del nombre que es fan servir i el temps computacional del mètode kNN dependrà també de la longitud de la finestra.

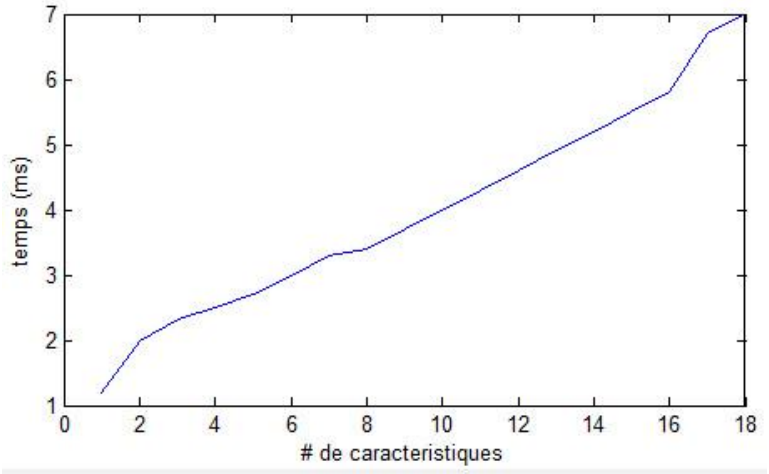


Figura 33. Cost del mètode kNN en funció del nombre de característiques

Tal com s'ha anat avançant en els capítols anteriors, la implementació de les funcions desenvolupades han tingut en ment aquest repte computacional.

La següent taula mostra el cost computacional del nombre màxim de funcions repetitives que es fan servir en la identificació per a una configuració de freqüència de mostreig de 22.050 Hz, 92 ms de longitud de finestra i utilització de totes les característiques. Tenint en compte que el encavalcament està implícit en el procés, s'ha de considerar un temps llindar de processament de 46 ms. La CPU de l'ordinador és Intel i7a 2GHz i per als prendre nota dels càlculs es fa servir les comandes tic i toc de Matlab:

Taula 19. Temps de computació amb 22.050 Hz i 92 ms de finestra

Funció	Temps
Filtre	0.0620 ms
RMS	0.0200 ms
ZCR	0.0620 ms

Enfinestrat i FFT	0.6070 ms
SC	0.6610 ms
SF	0.8040 ms
RO	0.8420 ms
12 MFCC	1.1890 ms
RMSRATIO	1.2570 ms
KNN	7.4950 ms
Total	**13.56 ms**

Es pot veure com el cost computacional total està molt per sota del llindar i, tal com es va comprovar a l'apartat 4.4.1. (Taula 8), correspon a una de les configuracions d'execució més exitoses (juntament amb la de 16.000 Hz – 128 ms).

La configuració de més cost computacional correspon a 22.050 Hz - 23 ms. Tot i que la finestra és més petita, el nombre de finestres a processar per segon s'incrementa i, a més, el repositori conté més mostres. En aquest, el cost computacional és:

Taula 20. Temps de computació amb 22.050 Hz i 23 ms de finestra

Funció	Temps
Filtre	0.052 ms
RMS	0.016 ms
ZCR	0.036 ms
Enfinestrat i FFT	0.353 ms
SC	0.401 ms
SF	0.491 ms
RO	0.531 ms
12 MFCC	0.692 ms
RMSRATIO	0.785 ms

KNN	27.978 ms
Total	**31 ms**

Tot i que les fases de transició tindrien un cost de només 0.068 ms (que serviria per a buidar memòria intermèdia), els resultats de la taula mostren com, amb la configuració 22.050 Hz – 23 ms, molt possiblement es perdria informació d'àudio ja que els 31 ms de temps de processament superen el llindar de 23 / 2 = 11.5 ms.

Vist això, i a menys que es facin servir combinacions de freqüència de mostreig molt altes amb longituds de finestra molt petites (combinacions, per altra banda no òptimes en quan a resultats) les funcions implementades permeten l'execució del sistema en temps real.

Per altra banda, combinacions menys exigents com 8.000 Hz – 128 ms segur que poden ser utilitzades per a execucions en temps real del sistema en ordinadors amb menys capacitat de computació.

CAPÍTOL 5 - INTERFÍCIE D'ÚS

5.1. INTRODUCCIÓ

Les eines desenvolupades permeten experimentar el sistema d'identificació de moviments respiratoris amb variades combinacions de freqüències de mostreig, longituds de finestres i conjunt de característiques acústiques simultànies que es volen utilitzar en l'experiment.

De les eines desenvolupades, i com assoliment de l'objectiu del Treball, s'han implementat dos versions del sistema d'identificació: una versió en mode consola (anomenada `asbslabconsole`) i una versió en entorn de finestres (anomenada `asbslab`). Totes dues sota el marc de treball de Matlab.

Els requeriments per a l'execució del sistema d'identificació en qualsevol dels modes són els següents:

- Matlab versió mínima 7.12.0 (R2011a). (Es desconeix si existeix incompatibilitats amb versions posteriors).
- DSP Toolbox de Matlab. S'utilitza únicament per a la captura d'àudio de forma asíncrona. Aquesta versió no permet, com si fa la versió del Matlab 2014, la detecció de saturació de la memòria intermèdia i conseqüent pèrdua d'informació.
- La carpeta `..\models\` on estan localitzats els models.

5.2. ASBSLABCONSOLE

`Asbslabconsole` permet executar el sistema d'identificació de forma directa des de la consola d'ordres del Matlab. Per a iniciar-lo, cal cridar la funció `asbslabconsole` amb els següents paràmetres:

Taula 21. Paràmetres d'execució d'asbslabconsole

Paràmetre		Descripció
Model	-	Codificació de la versió del model segons la Taula 4 .
Característiques	-	Conjunt de les característiques, segons la codificació de la Taula 5, que es volen aplicar i separades per el

		signe '+'. Exemple: ASBS_CENTROID+ASBS_MFCC3.
Arxiu	-	**(Opcional).** Si s'inclou aquest paràmetre, la identificació es realitza sobre l'arxiu passat per paràmetre. En cas contrari, la identificació es realitza sobre l'entrada d'àudio per defecte, normalment el micròfon.

5.2.1. Proves d'execució

Tot seguit figuren cinc exemples d'execució del sistema a través d'`asbslabconsole`. Per a indicar els moviments respiratoris que el sistema hagi identificat de manera errònia, aquests s'han enquadrat en el document.

> NOTA: Per a què Matlab les variables de les característiques, es procedeix a executar prèviament les següents funcions:
>
> ```
> EDU>> asbsinit(22050,92); asbsdispglobals
> ```

Exemple 1

Identificació dels moviments respiratoris de l'arxiu sample_c100s22.wav amb la versió del model s22w92 i basant-se només en el CENTROID com característica acústica.

```
EDU>> asbslabconsole('s22w92',ASBS_CENTROID,'..\gravacions\sample_c100s22')
Monitoring now...

mouth inspiration with 61.67 % of reliability
nasal espiration with 44.76 % of reliability
mouth espiration with 53.47 % of reliability
nasal espiration with 42.91 % of reliability
nasal inspiration with 46.27 % of reliability
nasal espiration with 49.30 % of reliability
mouth espiration with 37.84 % of reliability
nasal espiration with 50.74 % of reliability
EDU>>
```

Exemple 2

El mateix cas d'abans, però ara utilitzant només el RMSRatio i el tercer i quart coeficients MFCC:

```
EDU>>
asbslabconsole('s22w92',ASBS_MFCC3+ASBS_MFCC4+ASBS_RMSRATIO,'..\gravacions\sa
mple_c100s22')
Monitoring now...

mouth inspiration with 82.44 % of reliability
mouth espiration with 70.80 % of reliability
mouth inspiration with 85.39 % of reliability
mouth espiration with 67.19 % of reliability
nasal inspiration with 73.51 % of reliability
nasal espiration with 89.80 % of reliability
nasal inspiration with 70.44 % of reliability
nasal espiration with 63.05 % of reliability
EDU>>
```

Exemple 3

Identificació dels moviments respiratoris de l'arxiu sample_c100s16.wav amb la versió del model s16w96 i basant-se en tots els coeficients MFCC com característiques acústiques.

```
EDU>> asbslabconsole('s16w96',ASBS_MFCC,'..\gravacions\sample_c100s16')
Monitoring now...

mouth inspiration with 90.94 % of reliability
mouth espiration with 95.40 % of reliability
mouth inspiration with 94.90 % of reliability
mouth espiration with 94.91 % of reliability
nasal inspiration with 95.00 % of reliability
nasal espiration with 95.63 % of reliability
nasal inspiration with 94.71 % of reliability
nasal espiration with 96.38 % of reliability
EDU>>
```

Exemple 4

Identificació dels moviments respiratoris procedents de l'enregistrament en temps real pel micròfon amb la versió del model s22w92 i basant-se només en l'RMS Ratio i tercer i quart coeficients MFCC com característiques acústiques.

S'ha de tenir en compte que la prova es fa directament a través del propi micròfon de l'ordinador i sense accions especials d'aïllaments de sorolls externs. Són, per tant, unes condicions d'enregistrament diferents de les definides en l'escenari de treball. Per aquest fet, s'espera un pitjor índex d'identificacions positives.

```
EDU>> asbslabconsole('s22w92',ASBS_RMSRATIO+ASBS_MFCC3+ASBS_MFCC4)
Monitoring now... Press CTRL+C to stop.

nasal inspiration with 71.39 % of reliability
```

```
nasal espiration with 68.80 % of reliability
nasal inspiration with 73.05 % of reliability
mouth inspiration with 61.33 % of reliability
```

Exemple 5

Identificació dels moviments respiratoris procedents de l'enregistrament en temps real pel micròfon amb la versió del model s22w92 i basant-se en l'RMS Ratio i tots els coeficients MFCC com característiques acústiques. Igual que l'exemple anterior, la prova no es fa en les condicions d'enregistrament definides a l'escenari de treball. S'afegeix, a més, soroll addicional en certs moments.

```
EDU>> asbslabconsole('s22w92',ASBS_RMSRATIO+ASBS_MFCC)
Monitoring now... Press CTRL+C to stop.

Too much noise to detect breathing
Too much noise to detect breathing
mouth espiration with 84.30 % of reliability
nasal inspiration with 79.26 % of reliability
mouth espiration with 85.78 % of reliability
nasal inspiration with 95.47 % of reliability
mouth espiration with 87.01 % of reliability
nasal inspiration with 76.58 % of reliability
mouth espiration with 57.21 % of reliability
Too much noise to detect breathing
mouth espiration with 94.49 % of reliability
```

Exemple 6

Identificació dels moviments respiratoris procedents de l'enregistrament en temps real pel micròfon amb la versió del model s8w128 i basant-se en el Centroide, el MFCC3 i l'RMS Ratio (d'acord als resultat de la Taula 18. Igual que als exemples anteriors, la prova no es fa en les condicions d'enregistrament definides a l'escenari de treball.

```
EDU>> asbslabconsole('s8w128',ASBS_CENTROID+ASBS_MFCC3+ASBS_RMSRATIO)
Monitoring now... Press CTRL+C to stop.

nasal inspiration with 91.54 % of reliability
mouth espiration with 75.35 % of reliability
nasal inspiration with 95.00 % of reliability
mouth espiration with 53.83 % of reliability
mouth espiration with 81.37 % of reliability
mouth espiration with 68.43 % of reliability
mouth inspiration with 88.96 % of reliability
nasal espiration with 55.89 % of reliability
```

```
nasal inspiration with 95.43 % of reliability
nasal espiration with 57.25 % of reliability
mouth inspiration with 95.60 % of reliability
mouth espiration with 58.39 % of reliability??? Operation terminated by user
during ==> asbslabconsole at 136
```

5.3. ASBSLAB

ASBSLAB és la versió en entorn gràfic d'`asbslabconsole` amb les següents particularitats:

- Només permet l'enregistrament a través d'un dispositiu d'entrada d'àudio, ja sigui el micròfon, una línia auxiliar, etc. No permet utilitzar un arxiu com a font d'entrada.
- Permet la selecció del dispositiu d'entrada d'àudio entre els habilitats a l'ordinador.
- Incorpora un botó lliscant de guany d'entrada.

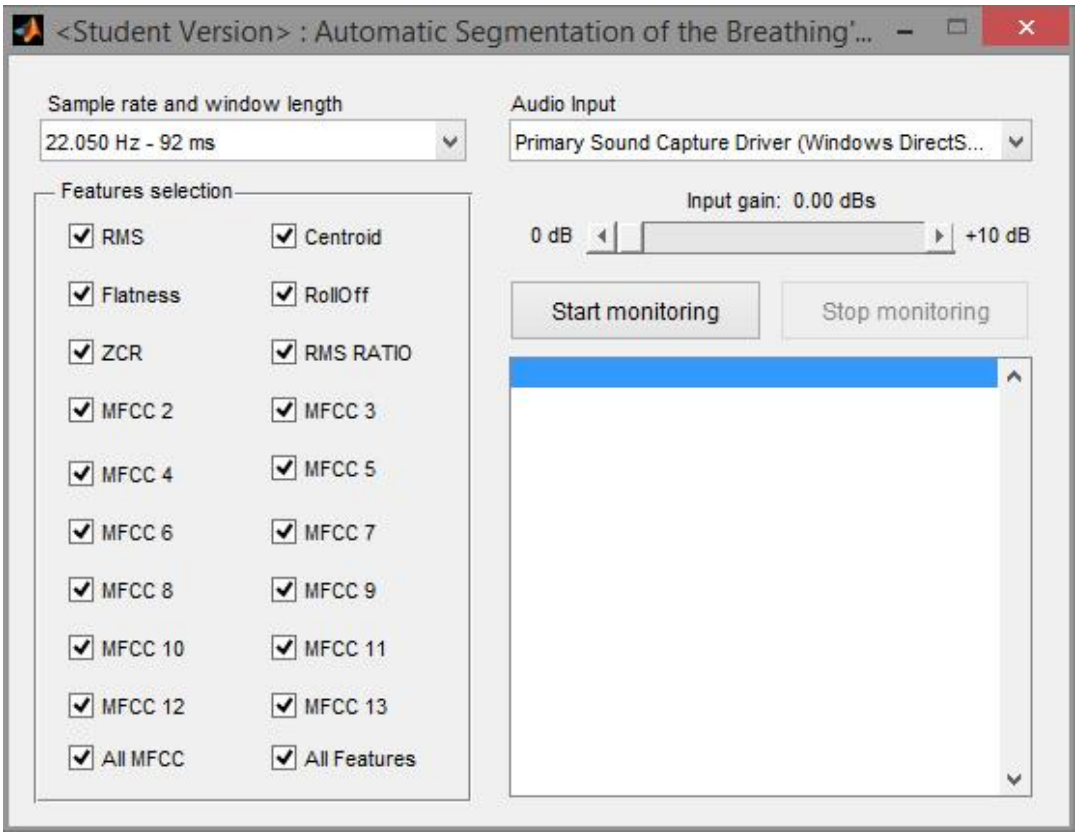


Figura 34. Interfície de l'asbslab

Amb el menú emergent d''Audio Input' s'escull el dispositiu d'entrada:

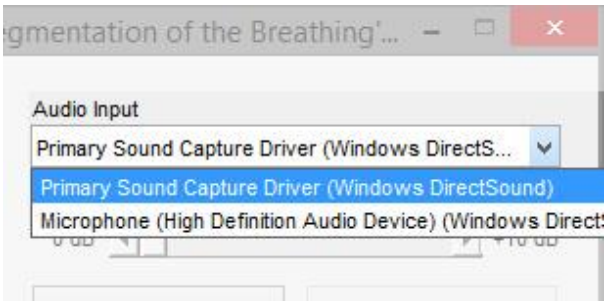


Figura 35. Selecció del dispositiu d'entrada

Amb el menú emergent 'Sample rate and window length', s'escull la configuració del model:

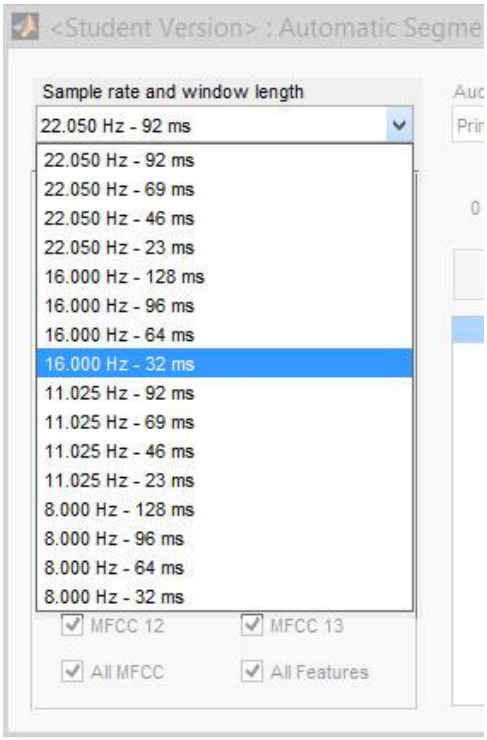


Figura 36. Selecció de la configuració del model de dades

Quan es prem el botó 'Start monitoring', el sistema comença a analitzar e identificar els moviments respiratoris a partir de la font d'àudio seleccionada:

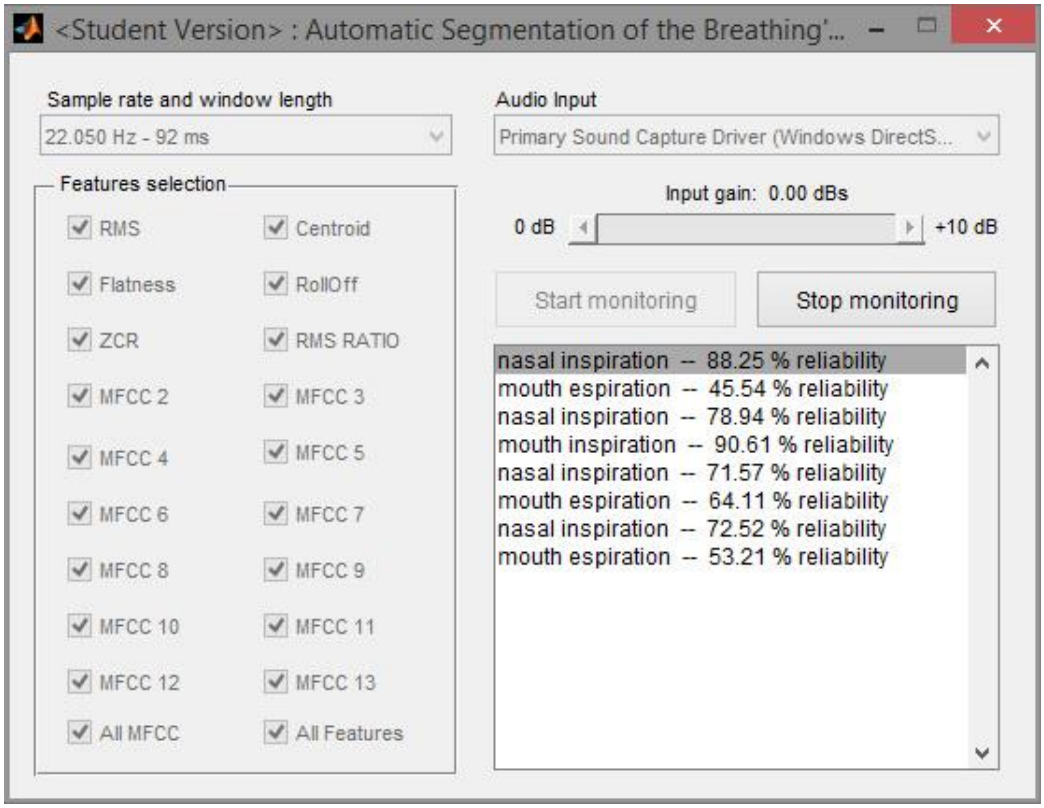


Figura 37. asbslab en execució

CAPÍTOL 6 - CONCLUSIONS

El propòsit d'aquest Treball Fi de Màster ha sigut la implementació d'una solució de programari per a identificar, en temps real, els moviments respiratoris -inspiració i espiració- a través d'un micròfon situat a una certa distància de l'usuari. L'objectiu s'ha assolit amb èxit i juntament amb la solució hi ha un conjunt de funcions ben estructurades que poden servir de base per a futurs projectes.

A l'apartat 6.1. es fa un resum general del procés de recerca i de desenvolupament de la solució, i treu conclusions dels resultats. L'apartat 6.2. proposa línies futures de treball.

6.1. VISIÓ GENERAL I CONCLUSIONS

Moltes vegades, un escenari de referència serveix per a incorporar dades inicials al projecte que són heretats d'estudis similars. Aquí no ha sigut el cas. Els escenaris i mètodes d'enregistrament trobats a la bibliografia, juntament amb els objectius dels autors, feia previsible que cap dada de processament es pogués donar per fet en aquest Treball. Aquí, per exemple, s'ha considerat adient, segons l'experimentació, descartar les freqüències per sota dels 100 Hz i permetre una amplada de banda d'enregistrament i processament fins als 22.050 Hz.

S'han analitzat i implementat l'extracció de 18 característiques acústiques (*RMS*, *spectrum centroid*, *spectrum flatness*, *rolloff*, *zerocross-rating*, *RatioRMS* i 12 coeficients MFCC). Les gràfiques de les distribucions dels valors de les característiques, aplicades als enregistraments ja processats dels moviments respiratoris dels voluntaris, ja avançaven la dificultat que podia significar la diferenciació entre inspiració i espiració a través d'una mateixa via respiratòria (nas o boca).

S'ha generat un repositori basat en més de 48 minuts d'enregistraments i que conformen un total de 1.784 moviments respiratoris. En funció de la freqüència de mostreig i longitud de finestra, els 1.784 moviments respiratoris s'han convertit en un nombre de mostres que va des de les 34.945 (8.000 Hz – 128 ms) a les 191.220 (22.050 Hz – 23 ms), mostres que formen el model de dades.

El sistema d'identificació s'ha implementat amb una combinació de dos mètodes. Primer de tot, s'ha utilitzat el mètode supervisat kNN per a classificar mostres desconegudes a partir del repositori. Posteriorment, s'ha implementat una màquina d'estats per a identificar el moviment respiratori en funció de la seqüència en el temps dels resultats de la classificació.

La validació del mètode classificació ha aportat les següents dades interessants:

- Els millors resultats s'aconsegueixen amb les combinacions 22.050 Hz – 92 ms i 16.000 Hz – 128 ms.
- La reducció de la freqüència de mostreig (a partir dels 16.000 Hz) fa decréixer el percentatge d'encerts, però d'una manera molt poc pronunciada i gairebé no perceptible si es considera totes les característiques (per exemple de 96.24 % amb 22.050 Hz a 96.22 % amb 8.000 Hz) .
- La reducció de la longitud de la finestra afecta de manera més pronunciada al percentatge d'encerts. Considerant totes les característiques, para cada freqüència de mostreig, les reduccions van del 96.64 % a 91.86 % (per a 22.050 Hz), de 96.87 % a 92.79 % (per a 16.000 Hz), de 96.51 % a 87.89 % (per a 11.025 Hz) i de 96.22 % a 89.07 % (per a 8.000 Hz).

En conseqüència, per a la identificació dels moviments respiratoris, reduir la freqüència de mostreig fins a 8.000 Hz no suposa un handicap, però la reducció de la longitud de finestra sí que influeix negativament de manera significativa en el percentatge d'encerts. Això, per altra banda és un avantatge si es té en compte el temps de computació, ja que com més petit és la longitud de finestra més alt és el temps de computació.

Si s'avalua el percentatge d'encerts quan es combinen de forma simultània un conjunt de característiques els resultats són interessants. S'han provat totes les combinacions possibles i, si bé era esperat que a mesura que s'incrementava el nombre de característiques creixia la mitjana de percentatges d'encerts, s'ha vist que el marge percentual d'encerts varia molt en funció de les característiques escollides. Dos exemples: 4 característiques ben seleccionades donen millor resultats que 10 mal escollides; per a 5 característiques, la diferencia entre la pitjor i millor combinació es de 25 %. S'ha elaborat una taula amb les millors combinacions de característiques, que poden servir de suggeriment en la configuració del sistema.

L'avaluació amb l'enregistrament d'un voluntari aliè a la generació del model de dades, dóna també valors positius. Arriba a un grau d'encert de quasi el 70 %.

La implementació de la màquina d'estats, ha fet pujar els percentatges d'encert a més del 93% per l'enregistrament d'un voluntari aliè a la generació del model de dades. Tot i això, per a arribar a aquests resultats és necessari implicar vàries característiques acústiques de manera simultània. La identificació del moviment respiratori amb l'ús de només una o dues característiques simultànies dóna uns resultats molt pobres.

Una qüestió important present en el desenvolupament de la solució ha estat el temps de computació. Un temps de computació per sobre de la meitat del temps de finestra significava que la solució no era viable per a ser executada en temps real. Per això, ha sigut molt important, estalviar càlculs repetitius que podien tractar-se com invariants una vegada escollida una configuració del model de dades, encara que sigui a força de perdre independència entre funcions. També s'ha tingut en compte el caràcter matricial de Matlab, on l'ús de bucles està molt penalitzat en cost computacional. Així, s'ha aconseguit un temps de computació de 13.56 ms per a una configuració etiquetada com a òptima (22.050 Hz – 92 ms) que requeria un màxim de 46 ms de cost computacional. Aquest ampli marge deixa espai a possibles ampliacions del sistema.

Finalment, la solució implementada permet l'experimentació amb diferents configuracions de freqüències de mostreig, longituds de finestra i conjunt de característiques, el que el fa adient per a continuar en la recerca.

Així mateix, es lliuren les eines per a ampliar, modificar o crear nous repositoris amb nous enregistraments.

6.2. LÍNIES FUTURES DE TREBALL

El propòsit d'aquest Treball Fi de Màster ha sigut implementació d'una solució per a identificar, en temps real, els moviments respiratoris -inspiració i espiració- a través d'un micròfon situat a una certa distància de l'usuari. Després d'haver assolit els reptes associats al Treball, i juntament amb els resultats aconseguits, es proposen les següents recomanacions sobre línies futures de treball:

- Millorar l'escenari d'enregistrament. Un mètode no invasiu, com l'utilitzat en aquest treball, és molt sensible a variacions a l'entorn que influeixen en la qualitat del sistema. Per altra banda, sistemes invasius tendeixen a ser incòmodes per l'usuari.

- S'ha demostrat que es pot aconseguir un sistema d'identificació amb un molt alt percentatge d'encerts amb un cost computacional tal que permeti pensar a implementar el sistema en dispositius de menys capacitat computacional com telèfons intel·ligents i tauletes. A més, l'exportació del codi Matlab a llenguatge C pot significar multiplicar per cinc la velocitat del sistema.
- El sistema identifica de manera correcta els moviments respiratoris. Igual que altres estudis, amb les mateixes funcions implementades, es podria afegir la identificació en temps real de més esdeveniments respiratoris com, per exemple, roncs.
- Matlab ha sigut una eina molt adient per a aquest treball. Malgrat tot, la versió utilitzada, R2011a, no incorpora tècniques alternatives d'aprenentatge computacional com Naive Bayes, Support Vector Machines (SVM) o Hidden Markov Models (HMM). Podria ser interessant l'avaluació en cost computacional i resultats del mètodes NB i SVM en substitució del kNN. Per altra banda, podria estudiar-se la viabilitat d'aplicació dels Hidden Markov Models com alternativa a la màquina d'estats elaborada de forma arbitrària.

BIBLIOGRAFIA

[1] Arnon Cohen and Dorota Landsberg, "Analysis and Automatic Classification of Breath Sounds," *Biomedical Engineering, IEEE Transactions on*, vol. BME-31, no. 9, pp. 584-590, 1984. [Online]. http://0-ieeexplore.ieee.org.cataleg.uoc.edu/stamp/stamp.jsp?tp=&arnumber=4121905

[2] L.P. Malmberg, G. Charbonneau, J. Vanderschoot A.R.A. Sovijärvi, "Characteristics of breath sounds and adventitious respiratory," *European Respiratory Review*, vol. 10, pp. 591-596, 2000.

[3] Theodoros Petsatodis, Christos Boukis, Ilias Maglogiannis Charalampos Doukas, "Automated sleep breath disorders detection utilizing patient sound analysis," *Biomedical Signal Processing and Control*, pp. 256-264, 2012. [Online]. http://dx.doi.org/10.1016/j.bspc.2012.03.002

[4] R.L. Moedomo, M.S. Mardiyanto, M. Ahmad, B. Alisjahbana, and T. Djatmiko, "The breath sound analysis for diseases diagnosis and stress measurement," *System Engineering and Technology (ICSET), 2012 International Conference on*, pp. 1-6, 2012. [Online]. http://0-ieeexplore.ieee.org.cataleg.uoc.edu/stamp/stamp.jsp?tp=&arnumber=6339358

[5] M. Oud and E.H. Dooijes, "Automated breath sound analysis," *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE*, vol. 3, pp. 990-992, 1996. [Online]. http://0-ieeexplore.ieee.org.cataleg.uoc.edu/stamp/stamp.jsp?tp=&arnumber=652675

[6] Divya S. Avalur, "Human Breath Detection using a Microphone," M.S. Thesis, Faculty of mathematics and natural science, Univ. of Groningen, Groningen, NL, 2013. [Online]. http://www.cs.rug.nl/~aiellom/tesi/avalur

[7] Brian R. and Kain, Alexander Snider, "Automatic Classification of Breathing Sounds During Sleep," in *Proceedings of The 38th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vancouver, Canada: IEEE, 2013. [Online]. http://www.cslu.ogi.edu/~sniderb/download/pub/Snider2013-ICASSP.pdf

[8] Urvi Patel, "Computerized Respiratory Sound Analysis: An Exploration of Methods," Thesis, Department of Physics, CWRU School of Engineering, Cleveland, Ohio, 2011. [Online]. http://www.phys.cwru.edu/undergrad/Senior%20Projects/PreviousProjects/papers/papers2012/Patel_Cechner.pdf

[9] Laura Mason, "Signal Processing Methods for Non-Invasive Respiration Monitoring," Thesis, Dept. Eng. Sci., University of Oxford, Oxford, UK, 2002. [Online]. http://www.ibme.ox.ac.uk/research/biomedical-signal-processing-

instrumentation/prof-l-tarassenko/publications/pdf/laura-thesis.pdf

[10] H. Alshaer, A. Pandya, T.D. Bradley, and F. Rudzicz, "Subject independent identification of breath sounds components using multiple classifiers," *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pp. 3577 - 3581, 2014. [Online]. http://0-ieeexplore.ieee.org.cataleg.uoc.edu/stamp/stamp.jsp?tp=&arnumber=6854267

[11] Zahra Moussavi Azadeh Yadollahi, "Automatic breath and snore sounds classification from tracheal and ambient sound recordings," *Medical Engineering & Physics*, pp. 985-990, 2010. [Online]. http://dx.doi.org/10.1016/j.medengphy.2010.06.013

[12] Benjin Wang, Lei Miao, Huiying Dong, and Zeguang Zheng, "The Research of Lung Sound Signals Based on Cepstrum Analysis," *Biomedical Engineering and Biotechnology (iCBEB), 2012 International Conference on*, pp. 934-938, 2012. [Online]. http://0-ieeexplore.ieee.org.cataleg.uoc.edu/stamp/stamp.jsp?tp=&arnumber=6245275

[13] G. Charbonneau, A. Giordano, P. Helistö, J. Vanderschoot B.M.G. Cheetham, "Digitization of data for respiratory sound recordings," *European Respiratory Review*, vol. 10, pp. 621-624, 2000.

[14] Y. Nishida, T. Hori, T. Suehiro, and S. Hirai, "Monitoring of breath sound under daily environment by ceiling dome microphone," *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 3, pp. 1822-1829, 2000. [Online]. http://0-ieeexplore.ieee.org.cataleg.uoc.edu/stamp/stamp.jsp?tp=&arnumber=886374

[15] K., Yuan, P. and Coyle, D. Curran, "Using Acoustic Sensors to Discriminate between Nasal and Mouth Breathing," *International Journal of Bioinformatics Research and Applications*, vol. 7, no. 4, Sept-Dec 2011.

[16] Peter Harrington, *Machine Learning in Action*. United States: Manning Publications Co., 2012.

[17] J.E. Earis, P. Helistö+, B.M.G. Cheetham++, M. Rossi, A.R.A. Sovijärvi, J. Vanderschoo L. Vannuccini, "Capturing and preprocessing of respiratory sounds," *European Respiratory Review*, vol. 10, pp. 616-620, 2000.

[18] B.M.G. Cheetham J.E. Earis, "Current methods used for computerized respiratory," *European Respiratory Review*, vol. 10, pp. 586-590, 2000.

ANNEX A – CODI FONT

Aquest annex inclou el codi de totes les funcions implementades en Matlab pel desenvolupament del Treball a excepció d'aquelles la finalitat de les quals és la generació de gràfics per a la documentació del treball o altres funcions auxiliars.

```
1:  function asbsinit ( fs, ws )
2:  %%
3:  % ASBSINIT(FS,WS)
4:  %
5:  % Purpose:  Performs global inititializations. It must be called before
6:  %           using any other functions. It performs generic calculations
7:  %           so that it saves computation time.
8:  %
9:  % Inputs:   fs          - sample rate
10: %           ws          - window size
11: %
12: % Output:   nothing
13: %
14: % Author:   juan a. Castro
15: % Version:  1.0 jan/2015
16: %%
17:
18: %%
19: global ASBS_INITDONE;
20: ASBS_INITDONE = true;
21: %%
22:
23: %% FRAME PROCESSING VARIABLES
24: global ASBS_SAMPLING;       % sampling frequency
25: global ASBS_WINDOW_SIZE;    % window size (s)
26: global ASBS_OVERLAPPED;     % true or false
27: global ASBS_THRESHOLD;      % RMS threshold
28: ASBS_SAMPLING       = fs;
29: ASBS_WINDOW_SIZE    = ws/1000;
30: ASBS_OVERLAPPED     = true;
31: ASBS_THRESHOLD      = 0.00125;
32: %%
33:
34: %% PREFILTERING DATA
35: global ASBS_ELLIP_ORDER;    % order of elliptic filter
36: global ASBS_ELLIP_CUT;      % cut frequency of filter
37: global ASBS_ELLIP_ZEROS;    % zeros filter coefficients
38: global ASBS_ELLIP_POLES;    % poles filter coefficients
39: ASBS_ELLIP_CUT   = 100;
40: ASBS_ELLIP_ORDER = 6;
41: [ASBS_ELLIP_ZEROS ASBS_ELLIP_POLES] = ellip (ASBS_ELLIP_ORDER, ...
```

```
                                    0.1, 50, ...
                                  2 * ASBS_ELLIP_CUT / ASBS_SAMPLING, ...
                                    'high');
%%

%% FEATURES VARIABLES
global ASBS_RMS;              % RMS
global ASBS_CENTROID;         % Spectrum Centroid
global ASBS_FLATNESS;         % Spectrum Flatness
global ASBS_ROLLOFF;          % Spectrum RollOff;
global ASBS_ZCR;              % Signal Zero-cross rating
global ASBS_MFCC2;            % MFCC 2
global ASBS_MFCC3;            % MFCC 3
global ASBS_MFCC4;            % MFCC 4
global ASBS_MFCC5;            % MFCC 5
global ASBS_MFCC6;            % MFCC 6
global ASBS_MFCC7;            % MFCC 7
global ASBS_MFCC8;            % MFCC 8
global ASBS_MFCC9;            % MFCC 9
global ASBS_MFCC10;           % MFCC 10
global ASBS_MFCC11;           % MFCC 11
global ASBS_MFCC12;           % MFCC 12
global ASBS_MFCC13;           % MFCC 13
global ASBS_MFCC;             % All MFCCs
global ASBS_RMSRATIO;         % RMS ratio at 2000Hz
global ASBS_ALLFEATURES;      % ALL FEATURES
ASBS_RMS              = bin2dec('1');
ASBS_CENTROID         = bin2dec('10');
ASBS_FLATNESS         = bin2dec('100');
ASBS_ROLLOFF          = bin2dec('1000');
ASBS_ZCR              = bin2dec('10000');
ASBS_MFCC2            = bin2dec('100000');
ASBS_MFCC3            = bin2dec('1000000');
ASBS_MFCC4            = bin2dec('10000000');
ASBS_MFCC5            = bin2dec('100000000');
ASBS_MFCC6            = bin2dec('1000000000');
ASBS_MFCC7            = bin2dec('10000000000');
ASBS_MFCC8            = bin2dec('100000000000');
ASBS_MFCC9            = bin2dec('1000000000000');
ASBS_MFCC10           = bin2dec('10000000000000');
ASBS_MFCC11           = bin2dec('100000000000000');
ASBS_MFCC12           = bin2dec('1000000000000000');
ASBS_MFCC13           = bin2dec('10000000000000000');
ASBS_MFCC             = bin2dec('1111111111100000');
ASBS_RMSRATIO         = bin2dec('100000000000000000');
ASBS_ALLFEATURES      = bin2dec('111111111111111111');

% Column indexes in features matrix
global ASBS_I_RMS;
global ASBS_I_CENTROID;
global ASBS_I_FLATNESS;
global ASBS_I_ROLLOFF;
```

```
94:  global ASBS_I_ZCR;
95:  global ASBS_I_MFCC2;
96:  global ASBS_I_MFCC3;
97:  global ASBS_I_MFCC4;
98:  global ASBS_I_MFCC5;
99:  global ASBS_I_MFCC6;
100: global ASBS_I_MFCC7;
101: global ASBS_I_MFCC8;
102: global ASBS_I_MFCC9;
103: global ASBS_I_MFCC10;
104: global ASBS_I_MFCC11;
105: global ASBS_I_MFCC12;
106: global ASBS_I_MFCC13;
107: global ASBS_I_RMSRATIO;
108: global ASBS_TOTAL_FEATURES;
109: global ASBS_I_EVENT;          % event
110: ASBS_I_RMS          = 1;
111: ASBS_I_CENTROID     = 2;
112: ASBS_I_FLATNESS     = 3;
113: ASBS_I_ROLLOFF      = 4;
114: ASBS_I_ZCR          = 5;
115: ASBS_I_MFCC2        = 6;
116: ASBS_I_MFCC3        = 7;
117: ASBS_I_MFCC4        = 8;
118: ASBS_I_MFCC5        = 9;
119: ASBS_I_MFCC6        = 10;
120: ASBS_I_MFCC7        = 11;
121: ASBS_I_MFCC8        = 12;
122: ASBS_I_MFCC9        = 13;
123: ASBS_I_MFCC10       = 14;
124: ASBS_I_MFCC11       = 15;
125: ASBS_I_MFCC12       = 16;
126: ASBS_I_MFCC13       = 17;
127: ASBS_I_RMSRATIO     = 18;
128: ASBS_TOTAL_FEATURES = 18;
129: ASBS_I_EVENT        = 19;
130: %%
131:
132: %% MFCC PRE-CALCULATIONS
133: global ASBS_WINDOW_SAMPLES;
134: global ASBS_NFFT;
135: global ASBS_TRIFBANK;
136: global ASBS_DCT;
137: global ASBS_LIFTER;
138: ASBS_WINDOW_SAMPLES = floor(fs * ASBS_WINDOW_SIZE);
139: if ASBS_OVERLAPPED % Window samples must be even
140:     ASBS_WINDOW_SAMPLES = round(ASBS_WINDOW_SAMPLES / 2) * 2;
141: end
142: % REFER TO ASBSMFCC.M FOR LICENCE OF THE FOLLOWING CODE
143: ASBS_NFFT = 2^nextpow2(ASBS_WINDOW_SAMPLES);
144: % Handy inline function handles
145: % Forward and backward mel frequency warping
```

```
146: hz2mel = @( hz )( 1127*log(1+hz/700) );     % Hertz to mel warping func
147: mel2hz = @( mel )( 700*exp(mel/1127)-700 ); % mel to Hertz warping func
148: % Type III DCT matrix routine
149: dctm = @( N, M )( sqrt(2.0/M) * cos( repmat([0:N-1].',1,M) ...
150:                                     .* repmat(pi*([1:M]-0.5)/M,N,1) ) );
151: % Cepstral lifter routine
152: ceplifter = @( N, L )( 1+0.5*L*sin(pi*[0:N-1]/L) );
153: % Triangular filterbank with uniformly spaced filters on mel scale
154: M = 24; % number of filters in filterbank
155: N = 13; % number of cepstrals coeficients including el 0
156: R = [ 100 ASBS_SAMPLING/2 ];  % Freq range to consider for filterbank
157: K = ASBS_NFFT/2+1;            % length of the unique part of the FFT
158: ASBS_TRIFBANK = trifbank( M, K, R, ASBS_SAMPLING, hz2mel, mel2hz );
159: % size of H is M x K
160: % DCT matrix computation
161: ASBS_DCT = dctm( N, M );
162: % Cepstral lifter computation. Used to give coefficients similar
163: % magnitud order
164: L = 22;
165: ASBS_LIFTER = ceplifter( N, L );
166: %%
167:
168: %% KNN Algorithm Data
169: global ASBS_MODELCODE;
170: global ASBS_ZMODEL;
171: global ASBS_MU;
172: global ASBS_DELTA;
173: ASBS_MODELCODE  = '';
174: ASBS_ZMODEL     = [];
175: ASBS_MU         = [];
176: ASBS_DELTA      = [];
177:
178: %% EOF
```

```
1:  function asbsfilter ( filename, cut )
2:  %%
3:  % ASBSFILTER(FILENAME,CUT)
4:  %
5:  % Purpose:  apply high pass filter and
6:  %           saves results as a .wav file ending with cXX, where XX is
7:  %           the cut frequency.
8:  %
9:  % Inputs:   filename    - source wave file
10: %           cut         - cut frequency
11: %
12: % Output:   none
13: %
14: % Author:   juan a. Castro
15: % Version:  1.0 jan/2015
16: %%
17:
```

```
18:  [path name ext] = fileparts (filename);
19:  if isempty( ext )
20:      filename = [filename '.wav'];
21:  end
22:
23:  % read audio file
24:  [x fs nbits] = wavread(filename);
25:
26:  % Calculate zeros and poles of a elliptic filter
27:  [b a] = ellip (6, 0.1, 50, 2 * cut / fs, 'high');
28:  % apply filter
29:  y = filter (b, a, x);
30:
31:  pos = strfind(filename, '_');
32:
33:  if isempty(pos)
34:      filename = [strrep(filename, '.wav', ['_c' int2str(cut) '.wav'])];
35:  else
36:      filename = [filename(1:pos) 'c' int2str(cut) filename(pos+1:end)];
37:  end
38:
39:  % save filtered wave as new file
40:  wavwrite(y, fs, nbits, filename);
41:
42:  %% EOF
```

```
1:   function rms = asbssignalrms ( x )
2:   %%
3:   % ASBSSIGNALRMS(X)
4:   %
5:   % Purpose:  Calculates RMS signal x
6:   %
7:   % Inputs:   x       - the signal
8:   %
9:   % Output:   rms     - rms of signal x
10:  %
11:  % Author:   juan a. Castro
12:  % Version:  1.0 jan/2015
13:  %%
14:
15:  rms = norm(x) / sqrt(length(x));
16:
17:  %% EOF
```

```
1:   function SC = asbscentroid ( X )
2:   %%
3:   % ASBSCENTROID(X)
4:   %
5:   % Purpose:  Returns the spectral centroid of spectrum signal X
6:   %
```

```
% Inputs:   X        - Spectrum signal
%
% Output:   SC       - Spectrum Centroid
%
% Author:   juan a. Castro
% Version:  1.0 jan/2015
%%

magnitude = 0;
for n = 1:length(X)
    magnitude = magnitude + n * X(n);
end
SC = magnitude / sum(X);

%% EOF
```

```
function SF = asbsflatness (X)
%%
% ASBSFLATNESS(X)
%
% Purpose:  Returns spectral flatness from spectrum signal X
%
% Inputs:   X        - Spectrum signal
%
% Output:   SF       - Spectrum flatness
%
% Author:   juan a. Castro
% Version:  1.0 jan/2015
%%

SF = geomean(X) / mean(X);

%% EOF
```

```
function RO = asbsrolloff (X, r)
%%
% ASBSROLLOFF(X,R)
%
% Purpose:  returns the roll-off with ratio r
%
% Inputs:   X        - the spectrum of the signal
%           R        - the ratio energy
%
% Output:   RO is the roll-off
%
% Author:   juan a. Castro
% Version:  1.0 jan/2015
%%

num = 0;
```

```
total = sum(X);
for i=1:length(X)
    num = num + X(i);
    if (num >= r * total)
        break;
    end

end
RO = i;

%% EOF
```

```
function ZCR = asbszerocross (x)
%%
% ASBSZEROCROSS(X)
%
% Purpose:  returns the zero/crossing rate of signal x
%
% Inputs:   x    - the signal
%
% Output:   ZCR - the zero-crossing rate
%
% Author:   juan a. Castro
% Version:  1.0 jan/2015
%%

ZCR = sum(abs(diff(x >= 0))) / length(x);

%% EOF
```

```
function CC = asbsmfcc(X)
%%
% ASBSMFCC(X)
%
% Purpose:  Calculates 2 to 13 Mel Frequency Cepstral Coeficients from
%           spectrum signal X.
%
% Inputs:   modelcode   - Repository
%
% Author:   juan a. Castro based on Kamil Wojcicki works
%           (see note below)
% Version:  1.0 jan/2015
%
%Copyright (c) 2011, Kamil Wojcicki
%All rights reserved.

%Redistribution and use in source and binary forms, with or without
%modification, are permitted provided that the following conditions are
%met:
%
```

```
21:  %    * Redistributions of source code must retain the above copyright
22:  %      notice, this list of conditions and the following disclaimer.
23:  %    * Redistributions in binary form must reproduce the above
24:  %      copyright notice, this list of conditions and the following
25:  %      disclaimer in the documentation and/or other materials provided
26:  %      with the distribution
27:  %    * Neither the name of the University of Texas at Dallas nor the
28:  %      names of its contributors may be used to endorse or promote
29:  %      products derived from this software without specific prior
30:  %      written permission.
31:
32:  %THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
33:  %"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
34:  %LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
35:  %A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
36:  %OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
37:  %SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
38:  %LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
39:  %DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
40:  %THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
41:  %(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
42:  %OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
43:  %%
44:
45:  global ASBS_TRIFBANK;
46:  global ASBS_DCT;
47:  global ASBS_LIFTER;
48:  global ASBS_NFFT;
49:
50:  K = ASBS_NFFT/2+1;
51:
52:  % Filterbank application to unique part of the magnitude spectrum
53:  FBE = ASBS_TRIFBANK * X(1:K); % FBE( FBE<1.0 ) = 1.0; % apply mel floor
54:
55:  % Conversion of logFBEs to cepstral coefficients through DCT
56:  CC =  ASBS_DCT * log( FBE );
57:
58:  % Cepstral liftering gives liftered cepstral coefficients
59:  CC = diag( ASBS_LIFTER ) * CC; % ~ HTK's MFCCs
60:  CC = CC(2:13);
61:
62:  % EOF
```

```
1:   function output = asbsrmsratio (X, Hz)
2:   %%
3:   % ASBSRMSRATIO(X,HZ)
4:   %
5:   % Purpose:  Calculates ratio of RMS from total RMS at Hz frquency.
6:   %
7:   % Inputs:   X       - Spectrum signal
8:   %           HZ      - Cut frequency
```

```
%
% Output:   Output  - Percent
%
% Author:   juan a. Castro
% Version:  1.0 jan/2015
%%

global ASBS_SAMPLING;

bin = floor(Hz * length(X) / (ASBS_SAMPLING / 2));

output = sum (X(1:bin)) / sum (X(bin:end));

%% EOF
```

```
function [ features ] = asbsframefeatures( frame, feature )
%%
% ASBSFRAMEFEATURES(FRAME,FEATURE)
%
% Purpose:  returns value of chosen features from a frame
%
% Inputs:   frame       - frame to process
%           feature     - feature(s) to extract
%
% Output:   features    - vector with selectec features
%
% Author:   juan a. Castro
% Version:  1.0 jan/2015
%%

global ASBS_THRESHOLD;
global ASBS_NFFT;
global ASBS_CENTROID;
global ASBS_FLATNESS;
global ASBS_ROLLOFF;
global ASBS_ZCR;
global ASBS_MFCC;
global ASBS_RMSRATIO;
global ASBS_WINDOW_SAMPLES;

global ASBS_I_RMS;            % RMS
global ASBS_I_CENTROID;       % Spectrum Centroid
global ASBS_I_FLATNESS;       % Spectrum Flatness
global ASBS_I_ROLLOFF;        % Spectrum RollOff;
global ASBS_I_ZCR;            % Signal Zero-cross rating
global ASBS_I_MFCC2;
global ASBS_I_MFCC13;
global ASBS_I_RMSRATIO;
global ASBS_TOTAL_FEATURES;

% set output dimensions
```

```
37:  features = zeros (1, ASBS_TOTAL_FEATURES);
38:
39:  % Calculates frame energy
40:  e = asbssignalrms( frame );
41:
42:  % Allways obtain energy
43:  features(1, ASBS_I_RMS) = e;
44:
45:  % chech if we need to process the frame
46:
47:  if (e >= ASBS_THRESHOLD)
48:      %% FEATURES EXTRACTION
49:      % ZERO-CROSSING RATE
50:      if bitand( feature, ASBS_ZCR ) > 0
51:          features(1, ASBS_I_ZCR) = asbszerocross( frame );
52:      end
53:      if bitand ( feature, ASBS_CENTROID ) > 0 | ...
54:         bitand ( feature, ASBS_FLATNESS ) > 0 | ...
55:         bitand ( feature, ASBS_ROLLOFF ) > 0 | ...
56:         bitand ( feature, ASBS_MFCC ) > 0 | ...
57:         bitand ( feature, ASBS_RMSRATIO ) > 0
58:
59:          %% FREQUENCY DOMAIN FEATURES
60:  %        N = length(frame);
61:          % apply hamming window
62:          xh = frame.*hamming(ASBS_WINDOW_SAMPLES);
63:  %        NFFT = 2^nextpow2(N);       % length of FFT
64:          X1 = fft (xh, ASBS_NFFT);        % spectrum of X
65:          X = X1 / ASBS_WINDOW_SAMPLES;
66:          XX = 2*abs(X(1 : ASBS_NFFT / 2 + 1)); % Spectrum magnitud
67:
68:          % SPECTRAL CENTROID
69:          if bitand ( feature, ASBS_CENTROID ) > 0
70:              features(1, ASBS_I_CENTROID) = ...
71:                                  asbscentroid(XX) / length (XX);
72:          end
73:
74:          % SPECTRAL FLATNESS
75:          if bitand ( feature, ASBS_FLATNESS ) > 0
76:              features(1, ASBS_I_FLATNESS) = asbsflatness(XX);
77:          end
78:
79:          % SPECTRAL ROLLOFF
80:          if bitand ( feature, ASBS_ROLLOFF ) > 0
81:              features(1, ASBS_I_ROLLOFF) = ...
82:                                  asbsrolloff(XX, 0.85) / length(XX);
83:          end
84:
85:          % MFCC
86:          if bitand ( feature, ASBS_MFCC ) > 0
87:              features(1, ASBS_I_MFCC2:ASBS_I_MFCC13) = ...
88:                                  asbsmfcc(abs(X1))'; %% X1 !!
```

```
89:          end
90:
91:          % RMS RATIO AT 2000 Hz
92:          if bitand ( feature, ASBS_RMSRATIO ) > 0
93:              features(1, ASBS_I_RMSRATIO) = asbsrmsratio(XX, 2000);
94:          end
95:      end     % spectrum feature
96:  end % if energy > threshold
97:
98:  %% EOF
```

```
1:   function [ features timemarks events ] = asbsfeatures( breathfile,...
2:                                                 feature, marked )
3:   %%
4:   % ASBSFEATURES(BREATHFILE,WS,AWO,TH,FEATURE,MARKS)
5:   %
6:   % Purpose:  returns chosen features, time marks and event of each
7:   %           obtained frame. Time marks are available if related
8:   %           file exists.
9:   %
10:  % Inputs:   breathfile  - file with breathing
11:  %           feature     - feature(s) to extract
12:  %           marked      - true if exist timemark file
13:  %
14:  %
15:  % Output:   features    - matrix (totalwindows, ASBS_TOTAL_FEATURES)
16:  %                         with selectec features
17:  %           timemarks   - matrix (totalwindows, 2) with start end time
18:  %                         of each windowframe
19:  %           events      - vector (totalwindows, 1) with associated
20:  %                         event  according to marks
21:  %
22:  % Author:   juan a. Castro
23:  % Version:  1.0 jan/2015
24:  %%
25:
26:  global ASBS_SAMPLING;
27:  global ASBS_OVERLAPPED;
28:  global ASBS_THRESHOLD;
29:  global ASBS_WINDOW_SAMPLES;
30:  global ASBS_TOTAL_FEATURES;
31:
32:  % append extension if
33:  [path name ext] = fileparts (breathfile);
34:  if isempty( ext )
35:      breathfile = [breathfile '.wav'];
36:  end
37:
38:  % Read breath file
39:  [x, fs, nbits] = wavread(breathfile);
40:
```

```
41:  if fs ~= ASBS_SAMPLING
42:      disp('Sampling does not match global setting');
43:      return
44:  end
45:
46:  if marked
47:      % Read marks file
48:      str = strfind(breathfile,'_');
49:      marks = asbsimportxls([breathfile(1:str(1)-1) '.xlsx']);
50:  end
51:
52:  if ASBS_OVERLAPPED, wpf = 2; else wpf = 1; end
53:
54:  % number of points in window
55:  step = floor (ASBS_WINDOW_SAMPLES / wpf);
56:
57:  % loop x framed
58:  totalwindows = floor(length(x) / step);
59:
60:  % set output dimensions
61:  features        = zeros (totalwindows, ASBS_TOTAL_FEATURES);
62:  timemarks       = zeros (totalwindows, 2);
63:  events          = zeros (totalwindows, 1);
64:
65:  imark = 1;
66:  j = 1;
67:  for i = 0 : totalwindows - wpf
68:
69:      a = i * step + 1;
70:      b = i * step + ASBS_WINDOW_SAMPLES;
71:
72:      % Calculates frame energy
73:      e = asbssignalrms( x(a:b) );
74:
75:      % chech if we need to process the frame
76:      if (e >= ASBS_THRESHOLD)
77:
78:          if marked
79:              %% ASSIGN TIME MARKS
80:              timea = a / fs;
81:              timeb = b / fs;
82:              timemarks (j, 1) = timea;
83:              timemarks (j, 2) = timeb;
84:
85:              %% ASSIGN EVENT
86:              while marks (imark, 1) < timeb
87:                  imark = imark + 1;
88:              end
89:              if marks(imark, 1) == timeb
90:                  events(j) = 0;
91:              elseif timea < marks(imark - 1, 1)
92:                  events (j) = 0;
```

```
93:                 else
94:                     events (j) = marks(imark, 2);
95:                 end
96:             end
97:             features (j, :) = asbsframefeatures(x(a:b), feature);
98:
99:             j = j + 1;
100:
101:        end % if energy > threshold
102:
103:    end % window loop
104:
105:    features (j:end, :) = [];
106:    timemarks(j:end, :) = [];
107:    events   (j:end, :) = [];
108:
109:    %% EOF
```

```
1:    function [ratioHz1 ratioHz2 ratioHz3 ratioHz4] = ...
2:                                      asbsrmsratios ( breathfile )
3:    %%
4:    % ASBSRMSRATIOS(BREATHFILE)
5:    %
6:    % Purpose:  Auxiliary function that calculates RMS average in steps
7:    %           of 50 Hz frequency.
8:    %
9:    % Inputs:   BRATHFILE    - wave file
10:   %
11:   % Output:   RATIOHZ1     - average energy for every 50 HZ bin frequency
12:   %                          for bucal inspiration.
13:   %           RATIOHZ2     - average energy for every 50 HZ bin frequency
14:   %                          for bucal espiration.
15:   %           RATIOHZ1     - average energy for every 50 HZ bin frequency
16:   %                          for nasal inspiration.
17:   %           RATIOHZ2     - average energy for every 50 HZ bin frequency
18:   %                          for nasal espiration.
19:   %
20:   % Author:   juan a. Castro
21:   % Version:  1.0 jan/2015
22:   %%
23:
24:   global ASBS_SAMPLING;
25:   global ASBS_WINDOW_SAMPLES;
26:   global ASBS_OVERLAPPED;
27:   global ASBS_THRESHOLD;
28:
29:   % append extension if needed
30:   [path name ext] = fileparts (breathfile);
31:   if isempty( ext )
32:       breathfile = [breathfile '.wav'];
33:   end
```

```
34:
35:  % Read breath file
36:  [x, fs, nbits] = wavread(breathfile);
37:
38:  if fs ~= ASBS_SAMPLING
39:      disp('Sampling does not match global setting');
40:      return
41:  end
42:
43:  % Read marks file
44:  str = strfind(breathfile,'_');
45:  marks = asbsimportxls([breathfile(1:str(1)-1) '.xlsx']);
46:
47:  if ASBS_OVERLAPPED, wpf = 2; else wpf = 1; end
48:
49:  % number of points in window
50:  step = floor (ASBS_WINDOW_SAMPLES / wpf);
51:
52:  % loop x framed
53:  totalwindows = floor(length(x) / step);
54:
55:  % set output dimensions
56:  ratios1         = zeros (totalwindows, 1);
57:  ratios2         = zeros (totalwindows, 1);
58:  ratios3         = zeros (totalwindows, 1);
59:  ratios4         = zeros (totalwindows, 1);
60:
61:  L = floor(fs / 2 / 50);
62:
63:  ratioHz1 = zeros (L, 1);
64:  ratioHz2 = zeros (L, 1);
65:  ratioHz3 = zeros (L, 1);
66:  ratioHz4 = zeros (L, 1);
67:
68:  idx = 1;
69:  for fc = 0:50:fs/2
70:      j1 = 1;
71:      j2 = 1;
72:      j3 = 1;
73:      j4 = 1;
74:      imark = 1;
75:      for i = 0 : totalwindows - wpf
76:          a = i * step + 1;
77:          b = i * step + ASBS_WINDOW_SAMPLES;
78:          e = asbssignalrms(x(a:b));
79:          if (e >= ASBS_THRESHOLD)
80:              timea = a / fs;
81:              timeb = b / fs;
82:
83:              while marks (imark, 1) < timeb
84:                  imark = imark + 1;
85:              end
```

```
86:                 if marks(imark, 1) == timeb
87:                     event = 0;
88:                 elseif timea < marks(imark-1, 1)
89:                     event = 0;
90:                 else
91:                     event = marks(imark, 2);
92:                 end
93:                 XX = asbsspectrum(x(a:b));
94:                 f = fs / 2 / length(XX);
95:                 sample = floor(fc/f - 0.5);
96:                 switch event
97:                     case 1
98:                         ratios1(j1,1) = sum(XX(1:sample)) / sum(XX);
99:                         j1 =  j1 + 1;
100:                    case 2
101:                        ratios2(j2,1) = sum(XX(1:sample)) / sum(XX);
102:                        j2 =  j2 + 1;
103:                    case 3
104:                        ratios3(j3,1) = sum(XX(1:sample)) / sum(XX);
105:                        j3 =  j3 + 1;
106:                    case 4
107:                        ratios4(j4,1) = sum(XX(1:sample)) / sum(XX);
108:                        j4 =  j4 + 1;
109:                end
110:            end
111:        end
112:        ratioHz1 (idx, 1) = mean (ratios1(1:j1));
113:        ratioHz2 (idx, 1) = mean (ratios2(1:j2));
114:        ratioHz3 (idx, 1) = mean (ratios3(1:j3));
115:        ratioHz4 (idx, 1) = mean (ratios4(1:j4));
116:        idx = idx + 1;
117:    end
118:
119:    %% EOF
```

```
1:   function asbsgeneratemodel (filename, fs, ws)
2:   %%
3:   % ASBSGENERATEMODEL(FILENAME,FS,WS)
4:   %
5:   % Purpose:  Generates repository from filename according to fs and ws.
6:   %           The repository is an Excel file with all features from all
7:   %           frames taken from source wave file.
8:   %
9:   % Inputs:   filename    - Source wave file taken as a model
10:  %           fs          - sample frequency
11:  %           ws          - window size
12:  %
13:  % Output:   none
14:  %
15:  % Author:   juan a. Castro
16:  % Version:  1.0 jan/2015
```

```
17:  %%
18:
19:  global ASBS_ALLFEATURES;
20:
21:  filename = strrep(filename, '.wav', '');
22:
23:  switch (fs)
24:      case 22050
25:          filename = [filename 's22'];
26:      case 16000
27:          filename = [filename 's16'];
28:      case 11025
29:          filename = [filename 's11'];
30:      case 8000
31:          filename = [filename 's8'];
32:  end
33:
34:  asbsinit ( fs, ws);
35:
36:  [ features timemarks events ] = asbsfeatures (filename,...
37:                                      ASBS_ALLFEATURES, true);
38:
39:  MODEL = [features events];
40:
41:  switch (ws)
42:      case 128
43:          filename = [filename 'w128'];
44:      case 96
45:          filename = [filename 'w96'];
46:      case 92
47:          filename = [filename 'w92'];
48:      case 69
49:          filename = [filename 'w69'];
50:      case 64
51:          filename = [filename 'w64'];
52:      case 46
53:          filename = [filename 'w46'];
54:      case 32
55:          filename = [filename 'w32'];
56:      case 23
57:          filename = [filename 'w23'];
58:  end
59:
60:  MODEL (find(MODEL (:, 19) == 0), :) = [];
61:
62:  xlswrite ([filename '.xlsx'], MODEL);
63:
64:  %% EOF
```

```
1:  %%
2:  % Purpose:  Generates all models data from 'breathing' audio. Each
```

```
3:  %           combination of sampling frequency and window size
4:  %           need its own repository.
5:  %
6:  % Author:   juan a. Castro
7:  % Version:  1.0 jan/2015
8:  %%
9:  asbsgeneratemodel( '..\models\breathing_c100.wav', 22050,  92 );
10: asbsgeneratemodel( '..\models\breathing_c100.wav', 22050,  69 );
11: asbsgeneratemodel( '..\models\breathing_c100.wav', 22050,  46 );
12: asbsgeneratemodel( '..\models\breathing_c100.wav', 22050,  23 );
13: asbsgeneratemodel( '..\models\breathing_c100.wav', 16000, 128 );
14: asbsgeneratemodel( '..\models\breathing_c100.wav', 16000,  96 );
15: asbsgeneratemodel( '..\models\breathing_c100.wav', 16000,  64 );
16: asbsgeneratemodel( '..\models\breathing_c100.wav', 16000,  32 );
17: asbsgeneratemodel( '..\models\breathing_c100.wav', 11025,  92 );
18: asbsgeneratemodel( '..\models\breathing_c100.wav', 11025,  69 );
19: asbsgeneratemodel( '..\models\breathing_c100.wav', 11025,  46 );
20: asbsgeneratemodel( '..\models\breathing_c100.wav', 11025,  23 );
21: asbsgeneratemodel( '..\models\breathing_c100.wav',  8000, 128 );
22: asbsgeneratemodel( '..\models\breathing_c100.wav',  8000,  96 );
23: asbsgeneratemodel( '..\models\breathing_c100.wav',  8000,  64 );
24: asbsgeneratemodel( '..\models\breathing_c100.wav',  8000,  32 );
25:
26: %% EOF
```

```
1:  function [ numbers strings ] = asbsimportxls( filename )
2:  %%
3:  % ASBSIMPORTXLS(FILENAME)
4:  %
5:  % Purpose:  Imports Excel file.
6:  %
7:  % Inputs:   filename    - Source Excel file
8:  %
9:  % Output:   numbers     - matrix with numeric data from Excel file
10: %           String      - Matrix with string data from Excel file
11: %
12: % Author:   juan a. Castro
13: % Version:  1.0 jan/2015
14: %%
15: sheetName='Sheet1';
16:
17: [numbers, strings] = xlsread(filename, sheetName);
18:
19: %% EOF
```

```
1:  function asbsloadmodel( code )
2:  %%
3:  % ASBSLOADMODEL(CODE)
4:  %
5:  % Purpose:  Loads model repository and performs initializations.
```

```
6:  %
7:  % Inputs:   code    - Repository code
8:  %
9:  % Author:   juan a. Castro
10: % Version:  1.0 jan/2015
11: %%
12: global ASBS_MODELCODE;
13: global ASBS_ZMODEL;
14: global ASBS_MU;
15: global ASBS_DELTA;
16: global ASBS_TOTAL_FEATURES;
17: global ASBS_I_EVENT;
18:
19: switch ( code )
20:     case 's22w92'
21:         asbsinit( 22050, 92 );
22:     case 's22w69'
23:         asbsinit( 22050, 69 );
24:     case 's22w46'
25:         asbsinit( 22050, 46);
26:     case 's22w23'
27:         asbsinit (22050, 23);
28:     case 's16w128'
29:         asbsinit (16000, 128);
30:     case 's16w96'
31:         asbsinit (16000, 96);
32:     case 's16w64'
33:         asbsinit (16000, 64);
34:     case 's16w32'
35:         asbsinit(16000, 32);
36:     case 's11w92'
37:         asbsinit(11025, 92);
38:     case 's11w69'
39:         asbsinit(11025, 69);
40:     case 's11w46'
41:         asbsinit(11025, 46);
42:     case 's11w23'
43:         asbsinit (11025, 23);
44:     case 's8w128'
45:         asbsinit(8000, 128);
46:     case 's8w96'
47:         asbsinit (8000, 96);
48:     case 's8w64'
49:         asbsinit(8000, 64);
50:     case 's8w32'
51:         asbsinit(8000, 32);
52:     otherwise
53:         disp ('Model not expected');
54:         exit;
55: end
56:
57: ASBS_MODELCODE = code;
```

```
58:
59:  filename = ['..\models\breathing_c100' code '.xlsx'];
60:
61:  modeldata = asbsimportxls(filename);
62:  %% STANDARDISE
63:  [ ASBS_ZMODEL ASBS_MU ASBS_DELTA ] = ...
64:              zscore(modeldata(:,1:ASBS_TOTAL_FEATURES));
65:  ASBS_ZMODEL = [ASBS_ZMODEL modeldata(:,ASBS_I_EVENT)];
66:
67:  %% EOF
```

```
1:   function index = asbsfeatureindex ( features )
2:   %%
3:   % ASBSFEATUREINDEX(FEATURES)
4:   %
5:   % Purpose:  return matrix indexes for features
6:   %
7:   % Inputs:   features    - Chosen features
8:   %
9:   % Output:   index       - matrix indexes
10:  %
11:  % Author:   juan a. Castro
12:  % Version:  1.0 jan/2015
13:  %%
14:
15:  global ASBS_RMS;
16:  global ASBS_CENTROID;
17:  global ASBS_FLATNESS;
18:  global ASBS_ROLLOFF;
19:  global ASBS_ZCR;
20:  global ASBS_MFCC2;
21:  global ASBS_MFCC3;
22:  global ASBS_MFCC4;
23:  global ASBS_MFCC5;
24:  global ASBS_MFCC6;
25:  global ASBS_MFCC7;
26:  global ASBS_MFCC8;
27:  global ASBS_MFCC9;
28:  global ASBS_MFCC10;
29:  global ASBS_MFCC11;
30:  global ASBS_MFCC12;
31:  global ASBS_MFCC13;
32:  global ASBS_RMSRATIO;
33:
34:  global ASBS_I_RMS;
35:  global ASBS_I_CENTROID;
36:  global ASBS_I_FLATNESS;
37:  global ASBS_I_ROLLOFF;
38:  global ASBS_I_ZCR;
39:  global ASBS_I_MFCC2;
40:  global ASBS_I_MFCC3;
```

```
41:  global ASBS_I_MFCC4;
42:  global ASBS_I_MFCC5;
43:  global ASBS_I_MFCC6;
44:  global ASBS_I_MFCC7;
45:  global ASBS_I_MFCC8;
46:  global ASBS_I_MFCC9;
47:  global ASBS_I_MFCC10;
48:  global ASBS_I_MFCC11;
49:  global ASBS_I_MFCC12;
50:  global ASBS_I_MFCC13;
51:  global ASBS_I_RMSRATIO;
52:
53:  index = [];
54:
55:  if bitand( features, ASBS_RMS ) > 0
56:      index = [index ASBS_I_RMS];
57:  end
58:  if bitand (features, ASBS_CENTROID ) > 0
59:      index = [index ASBS_I_CENTROID];
60:  end
61:  if bitand (features, ASBS_FLATNESS) > 0
62:      index = [index ASBS_I_FLATNESS];
63:  end
64:  if bitand(features, ASBS_ROLLOFF) > 0
65:      index = [index ASBS_I_ROLLOFF];
66:  end
67:  if bitand (features, ASBS_ZCR) > 0
68:      index = [index ASBS_I_ZCR];
69:  end
70:  if bitand (features, ASBS_MFCC2) > 0
71:      index = [index ASBS_I_MFCC2];
72:  end
73:  if bitand (features, ASBS_MFCC3) > 0
74:      index = [index ASBS_I_MFCC3];
75:  end
76:  if bitand (features, ASBS_MFCC4) > 0
77:      index = [index ASBS_I_MFCC4];
78:  end
79:  if bitand (features, ASBS_MFCC5) > 0
80:      index = [index ASBS_I_MFCC5];
81:  end
82:  if bitand (features, ASBS_MFCC6) > 0
83:      index = [index ASBS_I_MFCC6];
84:  end
85:  if bitand (features, ASBS_MFCC7) > 0
86:      index = [index ASBS_I_MFCC7];
87:  end
88:  if bitand (features, ASBS_MFCC8) > 0
89:      index = [index ASBS_I_MFCC8];
90:  end
91:  if bitand (features, ASBS_MFCC9) > 0
92:      index = [index ASBS_I_MFCC9];
```

```
93:   end
94:   if bitand (features, ASBS_MFCC10) > 0
95:       index = [index ASBS_I_MFCC10];
96:   end
97:   if bitand (features, ASBS_MFCC11) > 0
98:       index = [index ASBS_I_MFCC11];
99:   end
100:  if bitand (features, ASBS_MFCC12) > 0
101:      index = [index ASBS_I_MFCC12];
102:  end
103:  if bitand (features, ASBS_MFCC13) > 0
104:      index = [index ASBS_I_MFCC13];
105:  end
106:  if bitand (features, ASBS_RMSRATIO) > 0
107:      index = [index ASBS_I_RMSRATIO];
108:  end
109:
110:  %% EOF
```

```
1:   function asbscreateknnobj( features )
2:   %%
3:   % ASBSCREATEKNNOBJ(FEATURES)
4:   %
5:   % Purpose:  Creates KNNObj from createns matlab function and stores in
6:   %           ASBS_KNNOBJ global variable
7:   %
8:   % Inputs:   features    - chosen features
9:   %
10:  % Output:   none
11:  %
12:  % Author:   juan a. Castro
13:  % Version:  1.0 jan/2015
14:  %%
15:
16:  global ASBS_KNNOBJ;
17:  global ASBS_ZMODEL;
18:
19:  IFeatures = asbsfeatureindex (features);
20:
21:  ASBS_KNNOBJ = createns(ASBS_ZMODEL(:,IFeatures));
22:
23:  %% EOF
```

```
1:   function output = asbsknnvalidate (modelcode, features )
2:   %%
3:   % ASBSKNNVALIDATE(MODELCODE,FEATURES)
4:   %
5:   % Purpose:  Auxiliary function that performs validation of a repository
6:   %           using kNN classification algorithm.
7:   %
```

```
8:  % Inputs:   modelcode   - Repository
9:  %           features    - Chosen features to use in classification
10: %
11: % Output:   Matrix with real class and predicted class.
12: %
13: % Author:   juan a. Castro
14: % Version:  1.0 jan/2015
15: %%
16:
17: global ASBS_I_EVENT;
18: global ASBS_ZMODEL;
19:
20: asbsloadmodel( modelcode );
21:
22: %% MAKE PARTITION
23: I = randperm(length(ASBS_ZMODEL));
24: I = I(1:round(length(ASBS_ZMODEL)/3));
25:
26: query = ASBS_ZMODEL(I,:); %%
27:
28: %% SOLO USAR PARA BRWATHINGTEST
29: %IDX = find(ASBS_ZMODEL(:,19) >= 10);
30: %query = ASBS_ZMODEL(IDX, :);
31: %query (:,19) = query (:,19)./10;
32: %% FIN SOLO USAR...
33: % eliminar los que sale evento no definido
34: query = query(query( :, ASBS_I_EVENT)  ~= 0, : );
35:
36: ASBS_ZMODEL(I,:)=[];
37:
38: %% SOLO USAR PARA BRWATHINGTEST
39: %ASBS_ZMODEL(IDX,:)=[];
40: %% FIN SOLO USAR...
41:
42: %% STANDARDISE IS NOT NECESSARY BECAUSE TEST DATA BELONGS TO MODEL
43: zquery = query;
44:
45: IFeatures = asbsfeatureindex (features);
46: [IDX d] = knnsearch(ASBS_ZMODEL(:,IFeatures), zquery(:,IFeatures));
47:
48: nmatches1 = length(find(       ASBS_ZMODEL(IDX,ASBS_I_EVENT)    ==...
49:                                     query(:,ASBS_I_EVENT)     ));
50: nmatches2 = length(find( ceil (ASBS_ZMODEL(IDX,ASBS_I_EVENT)/2) ==...
51:                                     ceil(query(:,ASBS_I_EVENT)/2)  ));
52: nmatches3 = length(find( rem  (ASBS_ZMODEL(IDX,ASBS_I_EVENT),2) ==...
53:                                     rem (query(:,ASBS_I_EVENT), 2) ));
54: percent1 = nmatches1 / length(query);
55: percent2 = nmatches2 / length(query);
56: percent3 = nmatches3 / length(query);
57:
58: disp(['same event: ' num2str(percent1*100) '%']);
59: disp(['boca/nas  : ' num2str(percent2*100) '%']);
```

```
60:  disp(['ins/esp   : ' num2str(percent3*100) '%']);
61:
62:  %% outputs
63:  output = [query(:,ASBS_I_EVENT) ASBS_ZMODEL(IDX,ASBS_I_EVENT)];
64:
65:  %% EOF
```

```
1:   function output = asbsknnvalidate3 (modelcode, features )
2:   %%
3:   % ASBSKNNVALIDATE3(MODELCODE,FEATURES)
4:   %
5:   % Purpose:  Auxiliary function that performs cross-validation of a
6:   %           repository
7:   %           using kNN classification algorithm.
8:   %
9:   % Inputs:   modelcode   - Repository
10:  %           features    - Chosen features to use in classification
11:  %
12:  % Output:   output      - matrix with real and predicted classes
13:  %
14:  % Author:   juan a. Castro
15:  % Version:  1.0 jan/2015
16:  %%
17:
18:  global ASBS_I_EVENT;
19:  global ASBS_ZMODEL;
20:
21:  asbsloadmodel( modelcode );
22:
23:  p = zeros(3,3);
24:  for i=1:3
25:  I = randperm(length(ASBS_ZMODEL));
26:  L = floor(length(ASBS_ZMODEL)/3);
27:
28:  %% MAKE PARTITION
29:  I = I((i-1)*L+1:i*L);
30:  query = ASBS_ZMODEL(I,:);
31:  % eliminar los que sale evento no definido
32:  query = query(query( :, ASBS_I_EVENT)  ~= 0, : );
33:  ASBS_ZMODEL(I,:)=[];
34:
35:  zquery = query;
36:
37:  IFeatures = asbsfeatureindex (features);
38:  [IDX d] = knnsearch(ASBS_ZMODEL(:,IFeatures), zquery(:,IFeatures));
39:
40:  nmatches1 = length(find(       ASBS_ZMODEL(IDX,ASBS_I_EVENT)    ==...
41:                                   query(:,ASBS_I_EVENT)     ));
42:  nmatches2 = length(find( ceil (ASBS_ZMODEL(IDX,ASBS_I_EVENT)/2) ==...
43:                                   ceil(query(:,ASBS_I_EVENT)/2)  ));
44:  nmatches3 = length(find( rem  (ASBS_ZMODEL(IDX,ASBS_I_EVENT),2) ==...
```

```
45:                                     rem (query(:,ASBS_I_EVENT), 2) ));
46:   percent1 = nmatches1 / length(query);
47:   percent2 = nmatches2 / length(query);
48:   percent3 = nmatches3 / length(query);
49:
50:   p(i, 1) = percent1;
51:   p(i, 2) = percent2;
52:   p(i, 3) = percent3;
53:
54:   end
55:   %% outputs
56:   output = [query(:,ASBS_I_EVENT) ASBS_ZMODEL(IDX,ASBS_I_EVENT)];
57:
58:   disp(['MEAN: ' num2str(mean (p)*100)]);
59:   disp(['STD : ' num2str(std (p)*100)]);
60:
61:   %% EOF
```

```
1:   function asbsstminit ()
2:   %%
3:   % ASBSSTMINIT()
4:   %
5:   % Purpose:  Initialization of prediction state machine
6:   %
7:   % Inputs:   None
8:   %
9:   % Output:   None
10:  %
11:  % Author:   juan a. Castro
12:  % Version:  1.0 jan/2015
13:  %%
14:
15:  %% STATE MACHINE
16:
17:  global ASBS_STM_STATE;
18:  global ASBS_STM_OLDENERGY;
19:  global ASBS_STM_OLDVALUE;
20:  global ASBS_STM_STARTTIME;
21:  global ASBS_STM_NC0S;
22:  global ASBS_STM_NCN0S;
23:  global ASBS_STM_N1S;
24:  global ASBS_STM_N2S;
25:  global ASBS_STM_N3S;
26:  global ASBS_STM_N4S;
27:  global ASBS_STM_NCCE;
28:  global ASBS_STM_NCDE;
29:  global ASBS_STM_MINIMUM_CONSECUTIVES;
30:  %%
31:
32:  global ASBS_WINDOW_SIZE;
33:
```

```
34: %% STATE MACHINE
35: ASBS_STM_OLDENERGY = 0;
36: ASBS_STM_STATE = 0;
37: ASBS_STM_NC0S = 0;
38: ASBS_STM_NCN0S = 0;
39: ASBS_STM_N1S = 0;
40: ASBS_STM_N2S = 0;
41: ASBS_STM_N3S = 0;
42: ASBS_STM_N4S = 0;
43: ASBS_STM_NCCE = 0;
44: ASBS_STM_NCDE = 0;
45: ASBS_STM_STARTTIME = 0;
46: ASBS_STM_OLDVALUE = -1;
47: ASBS_STM_MINIMUM_CONSECUTIVES = 1 / ASBS_WINDOW_SIZE;
48: %%
49:
50: %% EOF
```

```
1:  function [ ev p ] = asbsstmactions (value, energy, timeevent )
2:  %%
3:  % ASBSSTMACTIONS(VALUE,EBERGY,TIMEEVENT)
4:  %
5:  % Purpose:  This is the prediction state machine
6:  %
7:  % Inputs:   VALUE       - Prediction from KNN classification
8:  %           ENERGY      - Energy of frame
9:  %           TIMEEVENT   - Time of frame
10: %
11: % Output:   EV          - Prediction from state machine
12: %                         A value of 0 means that it is not final
13: %                         state.
14: %           P           - Calculated probability of success if final
15: %                         state.
16: %
17: % Author:   juan a. Castro
18: % Version:  1.0 jan/2015
19: %%
20:
21: global ASBS_STM_STATE;
22: global ASBS_STM_OLDENERGY;
23: global ASBS_STM_OLDVALUE;
24: global ASBS_STM_NC0S;
25: global ASBS_STM_NCN0S;
26: global ASBS_STM_N1S;
27: global ASBS_STM_N2S;
28: global ASBS_STM_N3S;
29: global ASBS_STM_N4S;
30: global ASBS_STM_NCCE;
31: global ASBS_STM_NCDE;
32: global ASBS_STM_MINIMUM_CONSECUTIVES;
33:
```

```
34:  oldNC0S  = ASBS_STM_NC0S;
35:  oldNCN0S = ASBS_STM_NCN0S;
36:
37:  % Do accumulatives if states <> 1 and event <> 0
38:  if ASBS_STM_STATE ~= 1 || ASBS_STM_OLDVALUE == 0 || ...
39:          (ASBS_STM_STATE == 1 && value ~= 0)
40:      switch (value)
41:          case 0
42:              if ASBS_STM_OLDVALUE == 0
43:                  ASBS_STM_NC0S = ASBS_STM_NC0S + 1;
44:              else
45:                  ASBS_STM_NC0S = 0;
46:              end
47:          case 1
48:              ASBS_STM_N1S = ASBS_STM_N1S + 1;
49:          case 2
50:              ASBS_STM_N2S = ASBS_STM_N2S + 1;
51:          case 3
52:              ASBS_STM_N3S = ASBS_STM_N3S + 1;
53:          case 4
54:              ASBS_STM_N4S = ASBS_STM_N4S + 1;
55:      end
56:
57:      if  energy > ASBS_STM_OLDENERGY;
58:          ASBS_STM_NCCE = ASBS_STM_NCCE + 1;
59:      else
60:          ASBS_STM_NCDE = ASBS_STM_NCDE + 1;
61:      end
62:      ASBS_STM_OLDENERGY = energy;
63:  end
64:
65:  if value ~= 0 && ASBS_STM_OLDVALUE ~= 0
66:      ASBS_STM_NCN0S = ASBS_STM_NCN0S + 1;
67:  else
68:      ASBS_STM_NCN0S = 0;
69:  end
70:
71:  p = 0;
72:  switch (ASBS_STM_STATE)
73:      case 0
74:          if value == 0
75:              ASBS_STM_STATE = 1;
76:          end
77:          ev = 0;
78:      case 1
79:          if value == 0 && ASBS_STM_OLDVALUE ~= 0
80:              if oldNCN0S > ASBS_STM_MINIMUM_CONSECUTIVES
81:                  %% prendre resolucion
82:                  total = ASBS_STM_N1S + ASBS_STM_N2S + ...
83:                      ASBS_STM_N3S + ASBS_STM_N4S;
84:                  p1 = ASBS_STM_N1S / total;
85:                  p2 = ASBS_STM_N2S / total;
```

```
                    p3 = ASBS_STM_N3S / total;
                    p4 = ASBS_STM_N4S / total;
                    pBucal = (ASBS_STM_N1S + ASBS_STM_N2S) / total;
                    pNasal = (ASBS_STM_N3S + ASBS_STM_N4S) / total;
                    pIns   = (ASBS_STM_N1S + ASBS_STM_N3S) / total;
                    pEsp   = (ASBS_STM_N2S + ASBS_STM_N4S) / total;
                    pIns2  = ASBS_STM_NCCE / ...
                        (ASBS_STM_NCDE + ASBS_STM_NCCE);
                    pEsp2  = ASBS_STM_NCDE / ...
                        (ASBS_STM_NCDE + ASBS_STM_NCCE);

                    pBucalIns = 0.70 * p1 + 0.10 * pBucal + ...
                                0.10 * pIns + 0.10 * pIns2;
                    pBucalEsp = 0.70 * p2 + 0.10 * pBucal + ...
                                0.10 * pEsp + 0.10 * pEsp2;
                    pNasalIns = 0.70 * p3 + 0.10 * pNasal + ...
                                0.10 * pIns + 0.10 * pIns2;
                    pNasalEsp = 0.70 * p4 + 0.10 * pNasal + ...
                                0.10 * pEsp + 0.10 * pEsp2;

                    maxp = max([pBucalIns pBucalEsp pNasalIns pNasalEsp]);
                    p = maxp;
                    switch maxp
                        case pBucalIns
                            maxp = 1;
                        case pBucalEsp
                            maxp = 2;
                        case pNasalIns
                            maxp = 3;
                        case pNasalEsp
                            maxp = 4;
                    end
                    %%%
                    ev = maxp;
                    asbsstminit();
                    ASBS_STM_STATE = 2;
                else
                    asbsstminit();
                    ev = 0;
                    ASBS_STM_STATE = 2;
                end
            else
                ASBS_STM_STATE = 1;
                ev = 0;
            end
        case 2
            ASBS_STM_STATE = 1;
            ev = 0;
end

ASBS_STM_OLDVALUE = value;

```

```
138: %% EOF
```

```
1:   function asbslabconsole( model, features, varargin )
2:   %%
3:   % ASBSLABCONSOLE(MODEL,FEATURES)
4:   %
5:   % Purpose:  Identifies breathing events from default Audio Input Device
6:   %           filename (included in varargin
7:   %
8:   % Inputs:   model       - repository version (specific fs and ws)
9:   %           features    - features taken into account
10:  %           varargin    - the wav file in case identification will use
11:  %                         the file instead of microphone
12:  %
13:  % Author:   juan a. Castro
14:  % Version:  1.0 jan/2015
15:  %%
16:
17:  global ASBS_SAMPLING;
18:  global ASBS_WINDOW_SAMPLES;
19:  global ASBS_OVERLAPPED;
20:  global ASBS_THRESHOLD;
21:  global ASBS_ZMODEL;
22:  global ASBS_I_EVENT;
23:  global ASBS_MU;
24:  global ASBS_DELTA;
25:  global ASBS_KNNOBJ;
26:  global ASBS_ELLIP_ZEROS;
27:  global ASBS_ELLIP_POLES;
28:
29:  asbsloadmodel( model );
30:
31:  if nargin == 2  % Uses mcrophone
32:      AR = dsp.AudioRecorder;
33:      AR.SampleRate = ASBS_SAMPLING;
34:      AR.NumChannels = 1;
35:  elseif nargin == 3 % Read a file
36:          % append extension if
37:      [path name ext] = fileparts (varargin{1});
38:      if isempty( ext )
39:          filename = [varargin{1} '.wav'];
40:      end
41:      if exist(filename, 'file') == false
42:          disp ('wav file does not exist');
43:          return;
44:      end
45:      AR = dsp.AudioFileReader(filename);
46:      if AR.SampleRate ~= ASBS_SAMPLING
47:          disp('Sample rate does nt match with model');
48:          return;
49:      end
```

```
50:       AR.OutputDataType = 'double';
51:   else
52:       disp ('Invalid number or input arguments');
53:       return;
54:   end
55:
56:   SPF = AR.SamplesPerFrame;
57:
58:   if ASBS_OVERLAPPED, wpf = 2; else wpf = 1; end
59:
60:   % number of points in window
61:   SS = ASBS_WINDOW_SAMPLES / wpf;
62:
63:   asbscreateknnobj(features);
64:   IFeatures = asbsfeatureindex (features);
65:
66:   asbsstminit();
67:
68:   buffer = zeros(4096,1);
69:
70:   if nargin == 3
71:   %% IDENTIFYING FROM A FILE
72:   disp('Monitoring now...');
73:       BI = 1;
74:       tic;
75:       while ~isDone(AR)
76:           while BI <= ASBS_WINDOW_SAMPLES
77:               buffer(BI:BI+SPF-1) = step(AR);
78:               BI = BI + SPF;
79:           end
80:           x = filter (ASBS_ELLIP_ZEROS, ASBS_ELLIP_POLES,...
81:                       buffer(1:ASBS_WINDOW_SAMPLES));
82:           % Calculates frame energy
83:           e = asbssignalrms( x(1:ASBS_WINDOW_SAMPLES) );
84:
85:           % chech if we need to process the frame
86:           if (e >= ASBS_THRESHOLD)
87:               % Get features
88:
89:               zquery = asbsframefeatures(x(1:ASBS_WINDOW_SAMPLES),...
90:                                          features);
91:               zquery = bsxfun(@rdivide, ...
92:                           bsxfun(@minus,zquery, ASBS_MU),ASBS_DELTA);
93:               IDX = knnsearch(ASBS_KNNOBJ, zquery(:,IFeatures));
94:               value = ASBS_ZMODEL(IDX,ASBS_I_EVENT);
95:           else
96:               value = 0;
97:               tic;
98:           end % if energy > threshold
99:           [ev p ] = asbsstmactions(value, e, toc);
100:          if toc > 4
101:              tic;
```

```
102:             fprintf ('\nToo much noise to detect breathing');
103:             ev = 0;
104:             asbsstminit();
105:         end
106:         if ev > 0
107:             switch ev
108:                 case 1
109:                     str = 'mouth inspiration';
110:                 case 2
111:                     str = 'mouth espiration';
112:                 case 3
113:                     str = 'nasal inspiration';
114:                 case 4
115:                     str = 'nasal espiration';
116:             end
117:             fprintf('\n%s with %.2f %% of reliability', str, p*100);
118:         end
119:
120:         buffer(1:BI-SS) = buffer(SS+1:BI);
121:         BI = BI - SS;
122:
123:     end
124:     fprintf('\n');
125:     release (AR);
126:
127:     return;
128: end
129:
130: %% IDENTIFYING FROM MICROPHONE
131: disp('Monitoring now... Press CTRL+C to stop.');
132: BI = 1;
133: tic;
134: while true
135:     while BI <= ASBS_WINDOW_SAMPLES
136:         buffer(BI:BI+SPF-1) = step(AR);
137:         BI = BI + SPF;
138:     end
139:
140:     x = filter (ASBS_ELLIP_ZEROS, ASBS_ELLIP_POLES, ...
141:                     buffer(1:ASBS_WINDOW_SAMPLES));
142:     % Calculates frame energy
143:     e = asbssignalrms( x(1:ASBS_WINDOW_SAMPLES) );
144:
145:     % chech if we need to process the frame
146:     if (e >= ASBS_THRESHOLD)
147:         % Get features
148:         zquery = asbsframefeatures(x(1:ASBS_WINDOW_SAMPLES),...
149:                                    features);
150:         zquery = bsxfun(@rdivide, ...
151:                         bsxfun(@minus,zquery, ASBS_MU),ASBS_DELTA);
152:         IDX = knnsearch(ASBS_KNNOBJ, zquery(:,IFeatures));
153:         value = ASBS_ZMODEL(IDX,ASBS_I_EVENT);
```

```
154:      else
155:          value = 0;
156:          tic;
157:      end % if energy > threshold
158:      [ev p ] = asbsstmactions(value, e, toc);
159:      if toc > 4
160:          tic;
161:          fprintf ('\nToo much noise to detect breathing');
162:          ev = 0;
163:      end
164:
165:      if ev > 0
166:          switch ev
167:              case 1
168:                  str = 'mouth inspiration';
169:              case 2
170:                  str = 'mouth espiration';
171:              case 3
172:                  str = 'nasal inspiration';
173:              case 4
174:                  str = 'nasal espiration';
175:          end
176:          fprintf('\n%s with %.2f %% of reliability', str, p*100);
177:      end
178:
179:      buffer(1:BI-SS) = buffer(SS+1:BI);
180:      BI = BI - SS;
181:
182:  end % window loop
183:  fprintf('\n');
184:
185:  release(AR);
186:
187:  %% EOF
```

```
1:    function varargout = asbslab(varargin)
2:    % ASBSLAB MATLAB code for asbslab.fig
3:    %      ASBSLAB, by itself, creates a new ASBSLAB or raises the existing
4:    %      singleton*.
5:    %
6:    %      H = ASBSLAB returns the handle to a new ASBSLAB or the handle to
7:    %      the existing singleton*.
8:    %
9:    %      ASBSLAB('CALLBACK',hObject,eventData,handles,...) calls the lcal
10:   %      function named CALLBACK in ASBSLAB.M with the given input args.
11:   %
12:   %      ASBSLAB('Property','Value',...) creates a new ASBSLAB or raises
13:   %      the existing singleton*.  Starting from the left, property value
14:   %      pairs are applied to the GUI before asbslab_OpeningFcn gets
15:   %      called.  An unrecognized property name or invalid value makes
16:   %      property application pushbuttonstop.  All inputs are passed to
```

```
17:  %      asbslab_OpeningFcn via varargin.
18:  %
19:  %      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
20:  %      one instance to run (singleton)".
21:  %
22:  % See also: GUIDE, GUIDATA, GUIHANDLES
23:
24:  % Edit the above text to modify the response to help asbslab
25:
26:  % Last Modified by GUIDE v2.5 28-Dec-2014 20:22:08
27:
28:  % Begin initialization code - DO NOT EDIT
29:  gui_Singleton = 1;
30:  gui_State = struct('gui_Name',       mfilename, ...
31:                     'gui_Singleton',  gui_Singleton, ...
32:                     'gui_OpeningFcn', @asbslab_OpeningFcn, ...
33:                     'gui_OutputFcn',  @asbslab_OutputFcn, ...
34:                     'gui_LayoutFcn',  [] , ...
35:                     'gui_Callback',   []);
36:  if nargin && ischar(varargin{1})
37:      gui_State.gui_Callback = str2func(varargin{1});
38:  end
39:
40:  if nargout
41:      [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
42:  else
43:      gui_mainfcn(gui_State, varargin{:});
44:  end
45:  % End initialization code - DO NOT EDIT
46:
47:
48:  % --- Executes just before asbslab is made visible.
49:  function asbslab_OpeningFcn(hObject, eventdata, handles, varargin)
50:  % This function has no output args, see OutputFcn.
51:  % hObject    handle to figure
52:  % eventdata  reserved - to be defined in a future version of MATLAB
53:  % handles    structure with handles and user data (see GUIDATA)
54:  % varargin   command line arguments to asbslab (see VARARGIN)
55:
56:  % Choose default command line output for asbslab
57:  handles.output = hObject;
58:
59:  % Update handles structure
60:  guidata(hObject, handles);
61:
62:  % UIWAIT makes asbslab wait for user response (see UIRESUME)
63:  %uiwait(handles.asbslab);
64:
65:  % Initial model
66:  h = waitbar(0,'Loading model data. Please wait...');
67:  asbsloadmodel ('s22w92');
68:  close(h);
```

```
69:    global ASBS_ALLFEATURES;
70:    asbsdata = struct ('SelectedFeatures', ASBS_ALLFEATURES, 'toggle', 0);
71:    set(hObject,'UserData',asbsdata);
72:
73:
74:    % --- Outputs from this function are returned to the command line.
75:    function varargout = asbslab_OutputFcn(hObject, eventdata, handles)
76:    % varargout  cell array for returning output args (see VARARGOUT);
77:    % hObject    handle to figure
78:    % eventdata  reserved - to be defined in a future version of MATLAB
79:    % handles    structure with handles and user data (see GUIDATA)
80:
81:    % Get default command line output from handles structure
82:    varargout{1} = handles.output;
83:
84:
85:    % --- Executes on selection change in popupmenuModel.
86:    function popupmenuModel_Callback(hObject, eventdata, handles)
87:    % hObject    handle to popupmenuModel (see GCBO)
88:    % eventdata  reserved - to be defined in a future version of MATLAB
89:    % handles    structure with handles and user data (see GUIDATA)
90:
91:    h = waitbar(0,'Loading model data. Please wait...');
92:    switch get (hObject, 'Value')
93:        case 1
94:            asbsloadmodel ('s22w92');
95:        case 2
96:            asbsloadmodel ('s22w69');
97:        case 3
98:            asbsloadmodel ('s22w46');
99:        case 4
100:           asbsloadmodel ('s22w23');
101:       case 5
102:           asbsloadmodel ('s16w128');
103:       case 6
104:           asbsloadmodel ('s16w96');
105:       case 7
106:           asbsloadmodel ('s16w64');
107:       case 8
108:           asbsloadmodel ('s16w32');
109:       case 9
110:           asbsloadmodel ('s11w92');
111:       case 10
112:           asbsloadmodel ('s11w69');
113:       case 11
114:           asbsloadmodel ('s11w46');
115:       case 12
116:           asbsloadmodel ('s11w23');
117:       case 13
118:           asbsloadmodel ('s8w128');
119:       case 14
120:           asbsloadmodel ('s8w96');
```

```
121:        case 15
122:            asbsloadmodel ('s8w64');
123:        case 16
124:            asbsloadmodel ('s8w32');
125:    end
126:    waitbar(1);
127:    close(h);
128:
129:
130:    % --- Executes during object creation, after setting all properties.
131:    function popupmenuModel_CreateFcn(hObject, eventdata, handles)
132:    % hObject    handle to popupmenuModel (see GCBO)
133:    % eventdata  reserved - to be defined in a future version of MATLAB
134:    % handles    empty - handles not created until after all CreateFcns
135:    %            called
136:
137:    % Hint: popupmenu controls usually have a white background on Windows.
138:    %       See ISPC and COMPUTER.
139:    if ispc && isequal(get(hObject,'BackgroundColor'), ...
140:                       get(0,'defaultUicontrolBackgroundColor'))
141:        set(hObject,'BackgroundColor','white');
142:    end
143:
144:
145:
146:
147:    % --- Executes on button press in checkboxRMS.
148:    function checkboxRMS_Callback(hObject, eventdata, handles)
149:    % hObject    handle to checkboxRMS (see GCBO)
150:    % eventdata  reserved - to be defined in a future version of MATLAB
151:    % handles    structure with handles and user data (see GUIDATA)
152:
153:    % Hint: get(hObject,'Value') returns toggle state of checkboxRMS
154:    global ASBS_RMS;
155:    global ASBS_ALLFEATURES;
156:    data = get (handles.asbslab, 'UserData');
157:    value = get(hObject, 'Value');
158:    if value == 0
159:        data.SelectedFeatures = bitand(data.SelectedFeatures,...
160:                                bitcmp(uint32(ASBS_RMS)));
161:        set (handles.checkboxALL, 'Value', 0);
162:    else
163:        data.SelectedFeatures = bitor(data.SelectedFeatures,...
164:                                      uint32(ASBS_RMS));
165:        if data.SelectedFeatures == ASBS_ALLFEATURES
166:            set (handles.checkboxALL, 'Value', 1);
167:        end
168:    end
169:    set (handles.asbslab, 'UserData', data);
170:
171:
172:
```

```
% --- Executes on button press in checkboxMFCC5.
function checkboxMFCC5_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxMFCC5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkboxMFCC5
global ASBS_MFCC5;
global ASBS_MFCC;
global ASBS_ALLFEATURES;
data = get (handles.asbslab, 'UserData');
value = get(hObject, 'Value');
if value == 0
    data.SelectedFeatures = bitand(data.SelectedFeatures, ...
                            bitcmp(uint32(ASBS_MFCC5)));
    set (handles.checkboxALL, 'Value', 0);
    set (handles.checkboxALLMFCC, 'Value', 0);

else
    data.SelectedFeatures = bitor(data.SelectedFeatures,...
                                  uint32(ASBS_MFCC5));
    if data.SelectedFeatures == ASBS_ALLFEATURES
        set (handles.checkboxALL, 'Value', 1);
    end
    if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
        set (handles.checkboxALLMFCC, 'Value', 1);
    end
end
set (handles.asbslab, 'UserData', data);


% --- Executes on button press in checkboxMFCC6.
function checkboxMFCC6_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxMFCC6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkboxMFCC6
global ASBS_MFCC6;
global ASBS_MFCC;
global ASBS_ALLFEATURES;
data = get (handles.asbslab, 'UserData');
value = get(hObject, 'Value');
if value == 0
    data.SelectedFeatures = bitand(data.SelectedFeatures,...
                            bitcmp(uint32(ASBS_MFCC6)));
    set (handles.checkboxALL, 'Value', 0);
    set (handles.checkboxALLMFCC, 'Value', 0);

else
    data.SelectedFeatures = bitor(data.SelectedFeatures,...
                                uint32(ASBS_MFCC6));
```

```
225:        if data.SelectedFeatures == ASBS_ALLFEATURES
226:            set (handles.checkboxALL, 'Value', 1);
227:        end
228:        if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
229:            set (handles.checkboxALLMFCC, 'Value', 1);
230:        end
231:    end
232:    set (handles.asbslab, 'UserData', data);
233:
234:
235:    % --- Executes on button press in checkboxMFCC8.
236:    function checkboxMFCC8_Callback(hObject, eventdata, handles)
237:    % hObject    handle to checkboxMFCC8 (see GCBO)
238:    % eventdata  reserved - to be defined in a future version of MATLAB
239:    % handles    structure with handles and user data (see GUIDATA)
240:
241:    % Hint: get(hObject,'Value') returns toggle state of checkboxMFCC8
242:    global ASBS_MFCC8;
243:    global ASBS_MFCC;
244:    global ASBS_ALLFEATURES;
245:    data = get (handles.asbslab, 'UserData');
246:    value = get(hObject, 'Value');
247:    if value == 0
248:        data.SelectedFeatures = bitand(data.SelectedFeatures,...
249:                                       bitcmp(uint32(ASBS_MFCC8)));
250:        set (handles.checkboxALL, 'Value', 0);
251:        set (handles.checkboxALLMFCC, 'Value', 0);
252:
253:    else
254:        data.SelectedFeatures = bitor(data.SelectedFeatures,...
255:                                       uint32(ASBS_MFCC8));
256:        if data.SelectedFeatures == ASBS_ALLFEATURES
257:            set (handles.checkboxALL, 'Value', 1);
258:        end
259:        if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
260:            set (handles.checkboxALLMFCC, 'Value', 1);
261:        end
262:    end
263:    set (handles.asbslab, 'UserData', data);
264:
265:
266:    % --- Executes on button press in checkboxMFCC11.
267:    function checkboxMFCC11_Callback(hObject, eventdata, handles)
268:    % hObject    handle to checkboxMFCC11 (see GCBO)
269:    % eventdata  reserved - to be defined in a future version of MATLAB
270:    % handles    structure with handles and user data (see GUIDATA)
271:
272:    % Hint: get(hObject,'Value') returns toggle state of checkboxMFCC11
273:    global ASBS_MFCC11;
274:    global ASBS_MFCC;
275:    global ASBS_ALLFEATURES;
276:    data = get (handles.asbslab, 'UserData');
```

```
277:  value = get(hObject, 'Value');
278:  if value == 0
279:      data.SelectedFeatures = bitand(data.SelectedFeatures,...
280:                                     bitcmp(uint32(ASBS_MFCC11)));
281:      set (handles.checkboxALL, 'Value', 0);
282:      set (handles.checkboxALLMFCC, 'Value', 0);
283:
284:  else
285:      data.SelectedFeatures = bitor(data.SelectedFeatures,...
286:                                     uint32(ASBS_MFCC11));
287:      if data.SelectedFeatures == ASBS_ALLFEATURES
288:          set (handles.checkboxALL, 'Value', 1);
289:      end
290:      if bitand(data.SelectedFeatures, ASBS_MFCC) ==  ASBS_MFCC
291:          set (handles.checkboxALLMFCC, 'Value', 1);
292:      end
293:  end
294:  set (handles.asbslab, 'UserData', data);
295:
296:
297:  % --- Executes on button press in checkboxMFCC9.
298:  function checkboxMFCC9_Callback(hObject, eventdata, handles)
299:  % hObject    handle to checkboxMFCC9 (see GCBO)
300:  % eventdata  reserved - to be defined in a future version of MATLAB
301:  % handles    structure with handles and user data (see GUIDATA)
302:
303:  % Hint: get(hObject,'Value') returns toggle state of checkboxMFCC9
304:  global ASBS_MFCC9;
305:  global ASBS_MFCC;
306:  global ASBS_ALLFEATURES;
307:  data = get (handles.asbslab, 'UserData');
308:  value = get(hObject, 'Value');
309:  if value == 0
310:      data.SelectedFeatures = bitand(data.SelectedFeatures, ...
311:                                     bitcmp(uint32(ASBS_MFCC9)));
312:      set (handles.checkboxALL, 'Value', 0);
313:      set (handles.checkboxALLMFCC, 'Value', 0);
314:
315:  else
316:      data.SelectedFeatures = bitor(data.SelectedFeatures,...
317:                                     uint32(ASBS_MFCC9));
318:      if data.SelectedFeatures == ASBS_ALLFEATURES
319:          set (handles.checkboxALL, 'Value', 1);
320:      end
321:      if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
322:          set (handles.checkboxALLMFCC, 'Value', 1);
323:      end
324:  end
325:  set (handles.asbslab, 'UserData', data);
326:
327:
328:  % --- Executes on button press in checkboxRMSRATIO.
```

```
function checkboxRMSRATIO_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxRMSRATIO (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkboxRMSRATIO
global ASBS_RMSRATIO;
global ASBS_ALLFEATURES;
data = get (handles.asbslab, 'UserData');
value = get(hObject, 'Value');
if value == 0
    data.SelectedFeatures = bitand(data.SelectedFeatures, ...
                                   bitcmp(uint32(ASBS_RMSRATIO)));
    set (handles.checkboxALL, 'Value', 0);
else
    data.SelectedFeatures = bitor(data.SelectedFeatures,...
                                   uint32(ASBS_RMSRATIO));
    if data.SelectedFeatures == ASBS_ALLFEATURES
        set (handles.checkboxALL, 'Value', 1);
    end
end
set (handles.asbslab, 'UserData', data);



% --- Executes on button press in checkboxMFCC7.
function checkboxMFCC7_Callback(hObject, eventdata, handles)
% hObject    handle to checkboxMFCC7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkboxMFCC7
global ASBS_MFCC7;
global ASBS_MFCC;
global ASBS_ALLFEATURES;
data = get (handles.asbslab, 'UserData');
value = get(hObject, 'Value');
if value == 0
    data.SelectedFeatures = bitand(data.SelectedFeatures,...
                                   bitcmp(uint32(ASBS_MFCC7)));
    set (handles.checkboxALL, 'Value', 0);
    set (handles.checkboxALLMFCC, 'Value', 0);

else
    data.SelectedFeatures = bitor(data.SelectedFeatures, ...
                                   uint32(ASBS_MFCC7));
    if data.SelectedFeatures == ASBS_ALLFEATURES
        set (handles.checkboxALL, 'Value', 1);
    end
    if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
        set (handles.checkboxALLMFCC, 'Value', 1);
    end
```

```
381:  end
382:  set (handles.asbslab, 'UserData', data);
383:
384:
385:  % --- Executes on button press in checkboxMFCC13.
386:  function checkboxMFCC13_Callback(hObject, eventdata, handles)
387:  % hObject    handle to checkboxMFCC13 (see GCBO)
388:  % eventdata  reserved - to be defined in a future version of MATLAB
389:  % handles    structure with handles and user data (see GUIDATA)
390:
391:  % Hint: get(hObject,'Value') returns toggle state of checkboxMFCC13
392:  global ASBS_MFCC13;
393:  global ASBS_MFCC;
394:  global ASBS_ALLFEATURES;
395:  data = get (handles.asbslab, 'UserData');
396:  value = get(hObject, 'Value');
397:  if value == 0
398:      data.SelectedFeatures = bitand(data.SelectedFeatures,...
399:                                     bitcmp(uint32(ASBS_MFCC13)));
400:      set (handles.checkboxALL, 'Value', 0);
401:      set (handles.checkboxALLMFCC, 'Value', 0);
402:
403:  else
404:      data.SelectedFeatures = bitor(data.SelectedFeatures,...
405:                                     uint32(ASBS_MFCC13));
406:      if data.SelectedFeatures == ASBS_ALLFEATURES
407:          set (handles.checkboxALL, 'Value', 1);
408:      end
409:      if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
410:          set (handles.checkboxALLMFCC, 'Value', 1);
411:      end
412:  end
413:  set (handles.asbslab, 'UserData', data);
414:
415:
416:  % --- Executes on button press in checkboxALLMFCC.
417:  function checkboxALLMFCC_Callback(hObject, eventdata, handles)
418:  % hObject    handle to checkboxALLMFCC (see GCBO)
419:  % eventdata  reserved - to be defined in a future version of MATLAB
420:  % handles    structure with handles and user data (see GUIDATA)
421:
422:  % Hint: get(hObject,'Value') returns toggle state of checkboxALLMFCC
423:  global ASBS_ALLFEATURES;
424:  global ASBS_MFCC;
425:  data = get (handles.asbslab, 'UserData');
426:  value = get(hObject, 'Value');
427:  if value == 0
428:      data.SelectedFeatures = bitand(data.SelectedFeatures,...
429:                                     bitcmp(uint32(ASBS_MFCC)));
430:      set (handles.checkboxALL, 'Value', value);
431:  else
432:      data.SelectedFeatures = bitor(data.SelectedFeatures,...
```

```
433:                                     uint32(ASBS_MFCC));
434:         if data.SelectedFeatures == ASBS_ALLFEATURES
435:             set (handles.checkboxALL, 'Value', 1);
436:         end
437:     end
438:     set (handles.asbslab, 'UserData', data);
439: 
440:     set (handles.checkboxMFCC2, 'Value', value);
441:     set (handles.checkboxMFCC3, 'Value', value);
442:     set (handles.checkboxMFCC4, 'Value', value);
443:     set (handles.checkboxMFCC5, 'Value', value);
444:     set (handles.checkboxMFCC6, 'Value', value);
445:     set (handles.checkboxMFCC7, 'Value', value);
446:     set (handles.checkboxMFCC8, 'Value', value);
447:     set (handles.checkboxMFCC9, 'Value', value);
448:     set (handles.checkboxMFCC10, 'Value', value);
449:     set (handles.checkboxMFCC11, 'Value', value);
450:     set (handles.checkboxMFCC12, 'Value', value);
451:     set (handles.checkboxMFCC13, 'Value', value);
452: 
453: 
454: 
455:     % --- Executes on button press in checkboxRollOff.
456:     function checkboxRollOff_Callback(hObject, eventdata, handles)
457:     % hObject    handle to checkboxRollOff (see GCBO)
458:     % eventdata  reserved - to be defined in a future version of MATLAB
459:     % handles    structure with handles and user data (see GUIDATA)
460: 
461:     % Hint: get(hObject,'Value') returns toggle state of checkboxRollOff
462:     global ASBS_ROLLOFF;
463:     global ASBS_ALLFEATURES;
464:     data = get (handles.asbslab, 'UserData');
465:     value = get(hObject, 'Value');
466:     if value == 0
467:         data.SelectedFeatures = bitand(data.SelectedFeatures,...
468:                                     bitcmp(uint32(ASBS_ROLLOFF)));
469:         set (handles.checkboxALL, 'Value', 0);
470:     else
471:         data.SelectedFeatures = bitor(data.SelectedFeatures,...
472:                                     uint32(ASBS_ROLLOFF));
473:         if data.SelectedFeatures == ASBS_ALLFEATURES
474:             set (handles.checkboxALL, 'Value', 1);
475:         end
476:     end
477:     set (handles.asbslab, 'UserData', data);
478: 
479: 
480: 
481:     % --- Executes on button press in checkboxMFCC10.
482:     function checkboxMFCC10_Callback(hObject, eventdata, handles)
483:     % hObject    handle to checkboxMFCC10 (see GCBO)
484:     % eventdata  reserved - to be defined in a future version of MATLAB
```

```
485:  % handles    structure with handles and user data (see GUIDATA)
486:
487:  % Hint: get(hObject,'Value') returns toggle state of checkboxMFCC10
488:  global ASBS_MFCC10;
489:  global ASBS_MFCC;
490:  global ASBS_ALLFEATURES;
491:  data = get (handles.asbslab, 'UserData');
492:  value = get(hObject, 'Value');
493:  if value == 0
494:      data.SelectedFeatures = bitand(data.SelectedFeatures,...
495:                                     bitcmp(uint32(ASBS_MFCC10)));
496:      set (handles.checkboxALL, 'Value', 0);
497:      set (handles.checkboxALLMFCC, 'Value', 0);
498:
499:  else
500:      data.SelectedFeatures = bitor(data.SelectedFeatures,...
501:                                    uint32(ASBS_MFCC10));
502:      if data.SelectedFeatures == ASBS_ALLFEATURES
503:          set (handles.checkboxALL, 'Value', 1);
504:      end
505:      if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
506:          set (handles.checkboxALLMFCC, 'Value', 1);
507:      end
508:  end
509:  set (handles.asbslab, 'UserData', data);
510:
511:
512:  % --- Executes on button press in checkboxMFCC4.
513:  function checkboxMFCC4_Callback(hObject, eventdata, handles)
514:  % hObject    handle to checkboxMFCC4 (see GCBO)
515:  % eventdata  reserved - to be defined in a future version of MATLAB
516:  % handles    structure with handles and user data (see GUIDATA)
517:
518:  % Hint: get(hObject,'Value') returns toggle state of checkboxMFCC4
519:  global ASBS_MFCC4;
520:  global ASBS_MFCC;
521:  global ASBS_ALLFEATURES;
522:  data = get (handles.asbslab, 'UserData');
523:  value = get(hObject, 'Value');
524:  if value == 0
525:      data.SelectedFeatures = bitand(data.SelectedFeatures,...
526:                                     bitcmp(uint32(ASBS_MFCC4)));
527:      set (handles.checkboxALL, 'Value', 0);
528:      set (handles.checkboxALLMFCC, 'Value', 0);
529:
530:  else
531:      data.SelectedFeatures = bitor(data.SelectedFeatures,...
532:                                    uint32(ASBS_MFCC4));
533:      if data.SelectedFeatures == ASBS_ALLFEATURES
534:          set (handles.checkboxALL, 'Value', 1);
535:      end
536:      if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
```

```
537:            set (handles.checkboxALLMFCC, 'Value', 1);
538:        end
539:    end
540:    set (handles.asbslab, 'UserData', data);
541:
542:
543:    % --- Executes on button press in checkboxFlatness.
544:    function checkboxFlatness_Callback(hObject, eventdata, handles)
545:    % hObject    handle to checkboxFlatness (see GCBO)
546:    % eventdata  reserved - to be defined in a future version of MATLAB
547:    % handles    structure with handles and user data (see GUIDATA)
548:
549:    % Hint: get(hObject,'Value') returns toggle state of checkboxFlatness
550:    global ASBS_FLATNESS;
551:    global ASBS_ALLFEATURES;
552:    data = get (handles.asbslab, 'UserData');
553:    value = get(hObject, 'Value');
554:    if value == 0
555:        data.SelectedFeatures = bitand(data.SelectedFeatures,...
556:                                       bitcmp(uint32(ASBS_FLATNESS)));
557:        set (handles.checkboxALL, 'Value', 0);
558:    else
559:        data.SelectedFeatures = bitor(data.SelectedFeatures,...
560:                                      uint32(ASBS_FLATNESS));
561:        if data.SelectedFeatures == ASBS_ALLFEATURES
562:            set (handles.checkboxALL, 'Value', 1);
563:        end
564:    end
565:    set (handles.asbslab, 'UserData', data);
566:
567:
568:
569:    % --- Executes on button press in checkboxZCR.
570:    function checkboxZCR_Callback(hObject, eventdata, handles)
571:    % hObject    handle to checkboxZCR (see GCBO)
572:    % eventdata  reserved - to be defined in a future version of MATLAB
573:    % handles    structure with handles and user data (see GUIDATA)
574:
575:    % Hint: get(hObject,'Value') returns toggle state of checkboxZCR
576:    global ASBS_ZCR;
577:    global ASBS_ALLFEATURES;
578:    data = get (handles.asbslab, 'UserData');
579:    value = get(hObject, 'Value');
580:    if value == 0
581:        data.SelectedFeatures = bitand(data.SelectedFeatures,...
582:                                       bitcmp(uint32(ASBS_ZCR)));
583:        set (handles.checkboxALL, 'Value', 0);
584:    else
585:        data.SelectedFeatures = bitor(data.SelectedFeatures,...
586:                                      uint32(ASBS_ZCR));
587:        if data.SelectedFeatures == ASBS_ALLFEATURES
588:            set (handles.checkboxALL, 'Value', 1);
```

```
589:         end
590:     end
591:     set (handles.asbslab, 'UserData', data);
592: 
593: 
594: 
595:     % --- Executes on button press in checkboxMFCC3.
596:     function checkboxMFCC3_Callback(hObject, eventdata, handles)
597:     % hObject    handle to checkboxMFCC3 (see GCBO)
598:     % eventdata  reserved - to be defined in a future version of MATLAB
599:     % handles    structure with handles and user data (see GUIDATA)
600: 
601:     % Hint: get(hObject,'Value') returns toggle state of checkboxMFCC3
602:     global ASBS_MFCC3;
603:     global ASBS_MFCC;
604:     global ASBS_ALLFEATURES;
605:     data = get (handles.asbslab, 'UserData');
606:     value = get(hObject, 'Value');
607:     if value == 0
608:         data.SelectedFeatures = bitand(data.SelectedFeatures,...
609:                                        bitcmp(uint32(ASBS_MFCC3)));
610:         set (handles.checkboxALL, 'Value', 0);
611:         set (handles.checkboxALLMFCC, 'Value', 0);
612: 
613:     else
614:         data.SelectedFeatures = bitor(data.SelectedFeatures,...
615:                                       uint32(ASBS_MFCC3));
616:         if data.SelectedFeatures == ASBS_ALLFEATURES
617:             set (handles.checkboxALL, 'Value', 1);
618:         end
619:         if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
620:             set (handles.checkboxALLMFCC, 'Value', 1);
621:         end
622:     end
623:     set (handles.asbslab, 'UserData', data);
624: 
625: 
626:     % --- Executes on button press in checkboxCentroid.
627:     function checkboxCentroid_Callback(hObject, eventdata, handles)
628:     % hObject    handle to checkboxCentroid (see GCBO)
629:     % eventdata  reserved - to be defined in a future version of MATLAB
630:     % handles    structure with handles and user data (see GUIDATA)
631: 
632:     % Hint: get(hObject,'Value') returns toggle state of checkboxCentroid
633:     global ASBS_CENTROID;
634:     global ASBS_ALLFEATURES;
635:     data = get (handles.asbslab, 'UserData');
636:     value = get(hObject, 'Value');
637:     if value == 0
638:         data.SelectedFeatures = bitand(data.SelectedFeatures,...
639:                                        bitcmp(uint32(ASBS_CENTROID)));
640:         set (handles.checkboxALL, 'Value', 0);
```

```
641:  else
642:      data.SelectedFeatures = bitor(data.SelectedFeatures,...
643:                                    uint32(ASBS_CENTROID));
644:      if data.SelectedFeatures == ASBS_ALLFEATURES
645:          set (handles.checkboxALL, 'Value', 1);
646:      end
647:  end
648:  set (handles.asbslab, 'UserData', data);
649:
650:
651:  % --- Executes on button press in checkboxMFCC2.
652:  function checkboxMFCC2_Callback(hObject, eventdata, handles)
653:  % hObject    handle to checkboxMFCC2 (see GCBO)
654:  % eventdata  reserved - to be defined in a future version of MATLAB
655:  % handles    structure with handles and user data (see GUIDATA)
656:
657:  % Hint: get(hObject,'Value') returns toggle state of checkboxMFCC2
658:  global ASBS_MFCC2;
659:  global ASBS_MFCC;
660:  global ASBS_ALLFEATURES;
661:  data = get (handles.asbslab, 'UserData');
662:  value = get(hObject, 'Value');
663:  if value == 0
664:      data.SelectedFeatures = bitand(data.SelectedFeatures,...
665:                                    bitcmp(uint32(ASBS_MFCC2)));
666:      set (handles.checkboxALL, 'Value', 0);
667:      set (handles.checkboxALLMFCC, 'Value', 0);
668:
669:  else
670:      data.SelectedFeatures = bitor(data.SelectedFeatures,...
671:                                    uint32(ASBS_MFCC2));
672:      if data.SelectedFeatures == ASBS_ALLFEATURES
673:          set (handles.checkboxALL, 'Value', 1);
674:      end
675:      if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
676:          set (handles.checkboxALLMFCC, 'Value', 1);
677:      end
678:  end
679:  set (handles.asbslab, 'UserData', data);
680:
681:
682:
683:  % --- Executes on button press in checkboxALL.
684:  function checkboxALL_Callback(hObject, eventdata, handles)
685:  % hObject    handle to checkboxALL (see GCBO)
686:  % eventdata  reserved - to be defined in a future version of MATLAB
687:  % handles    structure with handles and user data (see GUIDATA)
688:
689:  % Hint: get(hObject,'Value') returns toggle state of checkboxALL
690:  global ASBS_ALLFEATURES;
691:  data = get (handles.asbslab, 'UserData');
692:  value = get(hObject, 'Value');
```

```
693:  if value == 0
694:      data.SelectedFeatures = 0;
695:      set (handles.checkboxALL, 'Value', 0);
696:  else
697:      data.SelectedFeatures = ASBS_ALLFEATURES;
698:      set (handles.checkboxALL, 'Value', 1);
699:  end
700:  set (handles.asbslab, 'UserData', data);
701:
702:  set (handles.checkboxRMS, 'Value', value);
703:  set (handles.checkboxCentroid, 'Value', value);
704:  set (handles.checkboxFlatness, 'Value', value);
705:  set (handles.checkboxRollOff, 'Value', value);
706:  set (handles.checkboxZCR, 'Value', value);
707:  set (handles.checkboxRMSRATIO, 'Value', value);
708:  set (handles.checkboxMFCC2, 'Value', value);
709:  set (handles.checkboxMFCC3, 'Value', value);
710:  set (handles.checkboxMFCC4, 'Value', value);
711:  set (handles.checkboxMFCC5, 'Value', value);
712:  set (handles.checkboxMFCC6, 'Value', value);
713:  set (handles.checkboxMFCC7, 'Value', value);
714:  set (handles.checkboxMFCC8, 'Value', value);
715:  set (handles.checkboxMFCC9, 'Value', value);
716:  set (handles.checkboxMFCC10, 'Value', value);
717:  set (handles.checkboxMFCC11, 'Value', value);
718:  set (handles.checkboxMFCC12, 'Value', value);
719:  set (handles.checkboxMFCC13, 'Value', value);
720:  set (handles.checkboxALLMFCC, 'Value', value);
721:
722:
723:  % --- Executes on button press in checkboxMFCC12.
724:  function checkboxMFCC12_Callback(hObject, eventdata, handles)
725:  % hObject    handle to checkboxMFCC12 (see GCBO)
726:  % eventdata  reserved - to be defined in a future version of MATLAB
727:  % handles    structure with handles and user data (see GUIDATA)
728:
729:  % Hint: get(hObject,'Value') returns toggle state of checkboxMFCC12
730:  global ASBS_MFCC12;
731:  global ASBS_MFCC;
732:  global ASBS_ALLFEATURES;
733:  data = get (handles.asbslab, 'UserData');
734:  value = get(hObject, 'Value');
735:  if value == 0
736:      data.SelectedFeatures = bitand(data.SelectedFeatures,...
737:                                     bitcmp(uint32(ASBS_MFCC12)));
738:      set (handles.checkboxALL, 'Value', 0);
739:      set (handles.checkboxALLMFCC, 'Value', 0);
740:
741:  else
742:      data.SelectedFeatures = bitor(data.SelectedFeatures,...
743:                                    uint32(ASBS_MFCC12));
744:      if data.SelectedFeatures == ASBS_ALLFEATURES
```

```
745:            set (handles.checkboxALL, 'Value', 1);
746:        end
747:        if bitand(data.SelectedFeatures, ASBS_MFCC) == ASBS_MFCC
748:            set (handles.checkboxALLMFCC, 'Value', 1);
749:        end
750:    end
751:    set (handles.asbslab, 'UserData', data);
752:
753:
754:    % --- Executes on selection change in popupmenuAudioInput.
755:    function popupmenuAudioInput_Callback(hObject, eventdata, handles)
756:    % hObject    handle to popupmenuAudioInput (see GCBO)
757:    % eventdata  reserved - to be defined in a future version of MATLAB
758:    % handles    structure with handles and user data (see GUIDATA)
759:
760:
761:
762:
763:    % --- Executes during object creation, after setting all properties.
764:    function popupmenuAudioInput_CreateFcn(hObject, eventdata, handles)
765:    % hObject    handle to popupmenuAudioInput (see GCBO)
766:    % eventdata  reserved - to be defined in a future version of MATLAB
767:    % handles    empty - handles not created until after all CreateFcns
768:    %            called
769:
770:    % Hint: popupmenu controls usually have a white background on Windows.
771:    %       See ISPC and COMPUTER.
772:    if ispc && isequal(get(hObject,'BackgroundColor'),...
773:                              get(0,'defaultUicontrolBackgroundColor'))
774:        set(hObject,'BackgroundColor','white');
775:    end
776:
777:    info = audiodevinfo;
778:    str = {};
779:    L = length(info.input);
780:    if L > 0
781:        for i=1:L
782:            str = [ str info.input(i).Name ] ;
783:        end
784:    else
785:        str =  {'< No Audio Input device found >'};
786:    end
787:    set (hObject, 'String', str);
788:
789:
790:
791:    % --- Executes on slider movement.
792:    function sliderGain_Callback(hObject, eventdata, handles)
793:    % hObject    handle to sliderGain (see GCBO)
794:    % eventdata  reserved - to be defined in a future version of MATLAB
795:    % handles    structure with handles and user data (see GUIDATA)
796:
```

```
797:  value = get(hObject,'Value');
798:  str = sprintf ('Input gain: %2.2f dBs', value);
799:  set (findobj('Tag','textGain'), 'String', str);
800:
801:
802:  % --- Executes during object creation, after setting all properties.
803:  function sliderGain_CreateFcn(hObject, eventdata, handles)
804:  % hObject    handle to sliderGain (see GCBO)
805:  % eventdata  reserved - to be defined in a future version of MATLAB
806:  % handles    empty - handles not created until after all CreateFcns
807:  %            called
808:
809:  % Hint: slider controls usually have a light gray background.
810:  if isequal(get(hObject,'BackgroundColor'),...
811:                        get(0,'defaultUicontrolBackgroundColor'))
812:      set(hObject,'BackgroundColor',[.9 .9 .9]);
813:  end
814:  set (hObject, 'Value', 0);
815:  set (findobj('Tag','textGain'), 'String', 'Input gain:  0.00 dBs');
816:
817:
818:  % --- Executes on selection change in listboxActions.
819:  function listboxActions_Callback(hObject, eventdata, handles)
820:  % hObject    handle to listboxActions (see GCBO)
821:  % eventdata  reserved - to be defined in a future version of MATLAB
822:  % handles    structure with handles and user data (see GUIDATA)
823:
824:
825:  % --- Executes during object creation, after setting all properties.
826:  function listboxActions_CreateFcn(hObject, eventdata, handles)
827:  % hObject    handle to listboxActions (see GCBO)
828:  % eventdata  reserved - to be defined in a future version of MATLAB
829:  % handles    empty - handles not created until after all CreateFcns
830:  %            called
831:
832:  % Hint: listbox controls usually have a white background on Windows.
833:  %       See ISPC and COMPUTER.
834:  if ispc && isequal(get(hObject,'BackgroundColor'),...
835:                        get(0,'defaultUicontrolBackgroundColor'))
836:      set(hObject,'BackgroundColor','white');
837:  end
838:
839:
840:  % --- Executes on button press in pushbuttonStart.
841:  function pushbuttonStart_Callback(hObject, eventdata, handles)
842:  % hObject    handle to pushbuttonStart (see GCBO)
843:  % eventdata  reserved - to be defined in a future version of MATLAB
844:  % handles    structure with handles and user data (see GUIDATA)
845:
846:  %% READ AND VERIFY PARAMETERS
847:  data = get (handles.asbslab, 'UserData');
848:  features = data.SelectedFeatures;
```

```
849:  if features == 0
850:      errordlg('At least 1 feature must be selected','Error','modal');
851:      return;
852:  end
853:  val = get(handles.popupmenuAudioInput, 'Value');
854:  string_list = get(handles.popupmenuAudioInput,'String');
855:  audiodevice = string_list{val};
856:  if audiodevice(1) == '<'
857:      errordlg('No audio input device found','Error','modal');
858:      return;
859:  end
860:  value = get (handles.sliderGain, 'Value');
861:  gain = 10^(value/20);
862:  %%
863:  %% DISABLE ITEMS
864:  set (handles.popupmenuModel,    'Enable', 'Off');
865:  set (handles.checkboxRMS,       'Enable', 'Off');
866:  set (handles.checkboxCentroid,  'Enable', 'Off');
867:  set (handles.checkboxFlatness,  'Enable', 'Off');
868:  set (handles.checkboxRollOff,   'Enable', 'Off');
869:  set (handles.checkboxZCR,       'Enable', 'Off');
870:  set (handles.checkboxRMSRATIO,  'Enable', 'Off');
871:  set (handles.checkboxMFCC2,     'Enable', 'Off');
872:  set (handles.checkboxMFCC3,     'Enable', 'Off');
873:  set (handles.checkboxMFCC4,     'Enable', 'Off');
874:  set (handles.checkboxMFCC5,     'Enable', 'Off');
875:  set (handles.checkboxMFCC6,     'Enable', 'Off');
876:  set (handles.checkboxMFCC7,     'Enable', 'Off');
877:  set (handles.checkboxMFCC8,     'Enable', 'Off');
878:  set (handles.checkboxMFCC9,     'Enable', 'Off');
879:  set (handles.checkboxMFCC10,    'Enable', 'Off');
880:  set (handles.checkboxMFCC11,    'Enable', 'Off');
881:  set (handles.checkboxMFCC12,    'Enable', 'Off');
882:  set (handles.checkboxMFCC13,    'Enable', 'Off');
883:  set (handles.checkboxALLMFCC,   'Enable', 'Off');
884:  set (handles.checkboxALL,       'Enable', 'Off');
885:  set (handles.popupmenuAudioInput, 'Enable', 'Off');
886:  set (handles.sliderGain, 'Enable', 'Off');
887:  set (handles.pushbuttonStart, 'Enable', 'Off');
888:  %%
889:  %% ENABLE PUSHBUTTONSTOP
890:  set (handles.pushbuttonStop, 'Enable', 'On');
891:  %% SET TOGGLE TO 1
892:  data = get (handles.asbslab, 'UserData');
893:  data.toggle = 1;
894:  set (handles.asbslab, 'UserData', data);
895:  %% CREATE AudioRecorder Object
896:  % Takeout (Windows DirectSound) from Audio device string in order to
897:  % be assigned to dsp.audiorecorder devicename
898:  global ASBS_SAMPLING;
899:  audiodevice = strrep(audiodevice, ' (Windows DirectSound)', '');
900:  AR = dsp.AudioRecorder;
```

```
901:  AR.DeviceName = audiodevice;
902:  AR.SampleRate = ASBS_SAMPLING;
903:  AR.NumChannels = 1;
904:  %% START RECORDING
905:  global ASBS_ELLIP_ZEROS;
906:  global ASBS_ELLIP_POLES;
907:  global ASBS_WINDOW_SAMPLES;
908:  global ASBS_THRESHOLD;
909:  global ASBS_MU;
910:  global ASBS_DELTA;
911:  global ASBS_ZMODEL;
912:  global ASBS_KNNOBJ;
913:  global ASBS_I_EVENT;
914:  global ASBS_OVERLAPPED;
915:
916:  EventList = {};
917:  SPF = AR.SamplesPerFrame;
918:
919:  if ASBS_OVERLAPPED, wpf = 2; else wpf = 1; end
920:
921:  % number of points in window
922:  SS = ASBS_WINDOW_SAMPLES / wpf;
923:
924:  asbscreateknnobj(features);
925:  IFeatures = asbsfeatureindex (features);
926:
927:  asbsstminit();
928:
929:  buffer = zeros(4096,1);
930:  BI = 1;
931:  tic;
932:  while true
933:      %% CHECK IF WE NEED TO STOP
934:      drawnow;
935:      data = get (handles.asbslab, 'UserData');
936:      if data.toggle == 0
937:          release(AR);
938:          break;
939:       end
940:      %% END CHECK IF...
941:
942:      while BI <= ASBS_WINDOW_SAMPLES
943:          buffer(BI:BI+SPF-1) = step(AR);
944:          BI = BI + SPF;
945:      end
946:
947:      %% APPLY GAIN
948:      x = buffer(1:ASBS_WINDOW_SAMPLES).*gain;
949:
950:      x = filter (ASBS_ELLIP_ZEROS, ASBS_ELLIP_POLES, x);
951:      % Calculates frame energy
952:      e = asbssignalrms( x(1:ASBS_WINDOW_SAMPLES) );
```

```
953:
954:        % chech if we need to process the frame
955:        if (e >= ASBS_THRESHOLD)
956:            % Get features
957:            zquery = asbsframefeatures(x(1:ASBS_WINDOW_SAMPLES), features);
958:            zquery = bsxfun(@rdivide,...
959:                            bsxfun(@minus,zquery, ASBS_MU),ASBS_DELTA);
960:            IDX = knnsearch(ASBS_KNNOBJ, zquery(:,IFeatures));
961:            value = ASBS_ZMODEL(IDX,ASBS_I_EVENT);
962:        else
963:            tic;
964:            value = 0;
965:        end % if energy > threshold
966:        [ev p ] = asbsstmactions(value, e, toc);
967:        if toc > 4
968:            tic;
969:            str = 'Too much noise or not breathing data';
970:            EventList {end+1} = str;
971:            set(handles.listboxActions, 'String', EventList);
972:            ev = 0;
973:            asbsstminit();
974:        end
975:        if ev > 0
976:            switch ev
977:                case 1
978:                    str = 'mouth inspiration';
979:                case 2
980:                    str = 'mouth espiration';
981:                case 3
982:                    str = 'nasal inspiration';
983:                case 4
984:                    str = 'nasal espiration';
985:            end
986:            str = sprintf('\n%s  --  %.2f %% reliability', str, p*100);
987:            EventList {end+1} = str;
988:            set(handles.listboxActions, 'String', EventList);
989:            drawnow;
990:        end
991:
992:        buffer(1:BI-SS) = buffer(SS+1:BI);
993:        BI = BI - SS;
994:
995:    end % window loop
996:    %% END RECORDING
997:
998:
999:    % --- Executes on button press in pushbuttonStop.
1000:   function pushbuttonStop_Callback(hObject, eventdata, handles)
1001:   % hObject    handle to pushbuttonStop (see GCBO)
1002:   % eventdata  reserved - to be defined in a future version of MATLAB
1003:   % handles    structure with handles and user data (see GUIDATA)
1004:   %%
```

```
1005: %% ENABLE ITEMS
1006: set (handles.popupmenuModel,    'Enable', 'On');
1007: set (handles.checkboxRMS,       'Enable', 'On');
1008: set (handles.checkboxCentroid,  'Enable', 'On');
1009: set (handles.checkboxFlatness,  'Enable', 'On');
1010: set (handles.checkboxRollOff,   'Enable', 'On');
1011: set (handles.checkboxZCR,       'Enable', 'On');
1012: set (handles.checkboxRMSRATIO,  'Enable', 'On');
1013: set (handles.checkboxMFCC2,     'Enable', 'On');
1014: set (handles.checkboxMFCC3,     'Enable', 'On');
1015: set (handles.checkboxMFCC4,     'Enable', 'On');
1016: set (handles.checkboxMFCC5,     'Enable', 'On');
1017: set (handles.checkboxMFCC6,     'Enable', 'On');
1018: set (handles.checkboxMFCC7,     'Enable', 'On');
1019: set (handles.checkboxMFCC8,     'Enable', 'On');
1020: set (handles.checkboxMFCC9,     'Enable', 'On');
1021: set (handles.checkboxMFCC10,    'Enable', 'On');
1022: set (handles.checkboxMFCC11,    'Enable', 'On');
1023: set (handles.checkboxMFCC12,    'Enable', 'On');
1024: set (handles.checkboxMFCC13,    'Enable', 'On');
1025: set (handles.checkboxALLMFCC,   'Enable', 'On');
1026: set (handles.checkboxALL,       'Enable', 'On');
1027: set (handles.popupmenuAudioInput, 'Enable', 'On');
1028: set (handles.sliderGain, 'Enable', 'On');
1029: set (handles.pushbuttonStart, 'Enable', 'On');
1030: %%
1031: %% DISABLE PUSHBUTTONSTOP
1032: set (handles.pushbuttonStop, 'Enable', 'Off');
1033: %%
1034: %% SET TOGGLE TO 0
1035: data = get (handles.asbslab, 'UserData');
1036: data.toggle = 0;
1037: set (handles.asbslab, 'UserData', data);
1038:
1039:
1040: % --- Executes during object creation, after setting all properties.
1041: function pushbuttonStop_CreateFcn(hObject, eventdata, handles)
1042: % hObject    handle to pushbuttonStop (see GCBO)
1043: % eventdata  reserved - to be defined in a future version of MATLAB
1044: % handles    empty - handles not created until after all CreateFcns
1045: %            called
1046:
1047: set (hObject, 'Enable', 'Off');
1048:
1049: % --- Executes when user attempts to close asbslab.
1050: function asbslab_CloseRequestFcn(hObject, eventdata, handles)
1051: % hObject    handle to asbslab (see GCBO)
1052: % eventdata  reserved - to be defined in a future version of MATLAB
1053: % handles    structure with handles and user data (see GUIDATA)
1054:
1055: % Hint: delete(hObject) closes the figure
1056: data = get (handles.asbslab, 'UserData');
```

```
1057: if data.toggle == 1
1058:     msgbox ('Stop recordin before closing application');
1059: else
1060:     delete(hObject);
1061: end
1062:
```

VITA

Juan Castro Mayorgas va néixer a Barcelona. S'ha graduat en Enginyeria Tècnica d'Informàtica (especialitat en Sistemes) al 2.009, Enginyeria Tècnica de Telecomunicacions (especialitat Telemàtica) al 2.011, assoleix amb aquest Treball el Màster Interuniversitari d'Enginyeria de Telecomunicació i, en simultani, està cursant el Màster Universitari d'Enginyeria Informàtica que té previst finalitzar el proper any 2.016.

Professionalment, desenvolupa la seva carrera laboral a diverses empreses de l'àmbit de les TIC.

Més informació: www.linkedin.com/in/jalbertocastro

es.linkedin.com/in/jalbertocastro/