Συστήματα Πολυμέσων και Εικονική Πραγματικότητα
Εργασία 2018-2019

Χριστίνα Θεοδωρίδου - 8055
Απόστολος Φανάκης - 8261

17 Φεβρουαρίου 2019

Περιεχόμενα

1 Εισαγωγή

Στην εργασία αυτή παρουσιάζεται μια απλοποιημένη εκδοχή του κωδικοποιητή και αποκωδικοποιητή AAC. Στην παρούσα αναφορά, περιγράφεται ο τρόπος υλοποίησης των βαθμίδων στο MATLAB και ο τρόπος χρήσης των συναρτήσεων από τις οποίες αποτελούνται, μαζί με κάποια ενδεικτικά αποτελέσματα. Το αρχείο που χρησιμοποιήθηκε για τα πειράματα είναι το δοσμένο 'LicorDeCalandraca.wav', το οποίο είναι δειγματοληπτημένο στα 48KHz και έχει 2 κανάλια. Τέλος, κατά την εκτέλεση των προγραμμάτων χρησιμοποιούνται οι συναρτήσεις mdct4 και imdct4 οι οποίες έχουν ληφθεί από προτεινόμενη ιστοσελίδα. [1]

2 1ο Επίπεδο

Στο πρώτο επίπεδο υλοποιήσαμε τις βαθμίδες Sequence Segmentation Control και Filterbank. Επίσης, υλοποιήθηκαν και οι συναρτήσεις AACoder1, η οποία εκτελεί τις προηγούμενες ρουτίνες και αποθηκεύει τα αποτελέσματα σε ένα struct, και η αντίστροφή της iAACoder.

2.1 *function* frameType = SSC(frameT, nextFrameT, prevFrameType)

Η πρώτη συνάρτηση που υλοποιήθηκε αφορά την βαθμίδα Sequence Segmentation Control, και σκοπός της είναι να καθορίσει τον τύπο ενός frame. Το πρώτο όρισμά της είναι ένας πίνακας 2048x2 , που περιέχει 2 κανάλια ήχου στο χρόνο, του frame του οποίου τον τύπο θέλουμε να διαπιστώσουμε. Αντίστοιχα, το δεύτερο όρισμα περιέχει τα κανάλια του του αμέσως επόμενου frame ενώ το τρίτο, αφορά τον τύπο του αμέσως προηγούμενου frame, ο οποίος μπορεί να πάρει 1 από τις αλφαριθμητικές τιμές "OLS" (ONLY_LONG_SEQUENCE), "LSS" (LONG_START_SEQUENCE), "ESH" (EIGHT_SHORT_SEQUENCE) και "LPS" (LONG_STOP_SEQUENCE) , ανάλογα το περιεχόμενο του.

Τα frame τα οποία είναι στατικά χαρακτηρίζονται ως 'OLS' ενώ τα frames με διακυμάνσεις ως 'ESH'. Τα μεταβατικά παράθυρα μεταξύ αυτών των κύριων κατηγοριών είναι τα 'LSS' και 'LPS'. Ο τρόπος με τον οποίον γίνεται η κατηγοριοποίηση περιγράφεται αναλυτικά στο πρότυπο του AAC. Γενικά, με τα βήματα που ακολουθούνται εξάγεται αρχικά ένα συμπέρασμα για κάθε κανάλι ξεχωριστά και έπειτα με μια συνδυαστική λογική που επίσης περιγράφεται στο πρότυπο, γίνεται ο συνδυασμός των αποφάσεων για να προκύψει ο τελικός τύπος του παραθύρου.

2.2 *function* frameF = filterbank(frameT, frameType, winType)

Η συνάρτηση αυτή αφορά την βαθμίδα filterbank και έχει ως ορίσματα ένα frame ήχου 2048x2, τον τύπο του καθώς και τον τύπο παραθύρου που θα χρησιμοποιηθεί. Η δε έξοδός του, είναι η αναπαράσταση του ίδιου του frame στο πεδίο της συχνότητας, σε όρους MDCT. Στην περίπτωση που το frame είναι τύπου "ESH", η συνάρτηση θα επιστρέψει 8 υποπίνακες 128x2 με τους συντελεστές του MDCT, έναν για κάθε subframe ενώ σε άλλη περίπτωση, θα επιστρέψει έναν πίνακα 1024x2.

Για να γίνει η μετάβαση αυτή ακολουθείται η διαδικασία που περιγράφεται στο πρότυπο. Αρχικά, στην υλοποιημένη συνάρτηση, δημιουργούνται οι τύποι των παραθύρων "KBD" και "SIN" που θα χρησιμοποιηθούν αργότερα, σε εκδοχές short και long για subframes και frames μήκους 256 και 2048 αντίστοιχα. Στη συνέχεια, σύμφωνα με το frameType και winType που δίνονται, εφαρμόζεται στο frameT το κατάλληλο παράθυρο, με τον τρόπο που υποδεικνύεται στην εκφώνηση και καλείται η συνάρτηση mdct4 [2] έτσι ώστε να ληφθούν οι συντελεστές MDCT.

Στην περίπτωση που ο τύπος του frame είναι "ESH", πρώτα το χωρίζουμε σε 8 subframes και έπειτα εφαρμόζουμε την παραθυροποίηση και τον MDCT.

[1] http://www.ee.columbia.edu/m̃arios/mdct/mdct_giraffe.html
[2] http://www.ee.columbia.edu/m̃arios/mdct/mdct_giraffe.html

2.3 *function* frameT = iFilterbank(frameF, frameType, winType)

Η συνάρτηση της iFilterbank, αποτελεί την αντίστροφη της filterbank. Δέχεται, δηλαδή, σαν όρισμα τους συντελεστές MDCT ενός frame, τον τύπο του και τον τύπο του παραθύρου που θα χρησιμοποιηθεί ενώ ως έξοδο, επιστρέφει έναν πίνακα 2048x2 που αντιστοιχεί στο frame στο πεδίο του χρόνου.

Αντίστοιχα με την filterbank, πρώτα ορίζονται τα παράθυρα “KBD” και “SIN”. Πριν εφαρμοστούν, όμως, κατάλληλα στο frame, εφαρμόζουμε την συνάρτηση imdct4 για να επιστρέψουμε στο πεδίο του χρόνου. Στην περίπτωση που ο τύπος τους είναι “ESH”, πρώτα εφαρμόζεται η συνάρτηση imdct4 σε όλα τα subframes, έπειτα γίνεται η παραθυροποίησή τους και στο τέλος αθροίζονται με κατάλληλο τρόπο ώστε στην έξοδο να επιστραφεί ολόκληρο το frame.

2.4 *function* AACSeq1 = AACoder1(fNameIn)

Για να εφαρμοστούν οι παραπάνω βαθμίδες, υλοποιήσαμε τον κωδικοποιητή AACoder1, ο οποίος παίρνει σαν μοναδικό όρισμα το path του WAV αρχείου που επιθυμούμε να κωδικοποιήσουμε, το οποίο πρέπει να είναι δειγματοληπτημένο στα 48KHz και να έχει 2 κανάλια, ενώ σαν έξοδο επιστρέφει ένα struct με διάσταση k x 1, όπου k, ο αριθμός των frames του αρχείου.

Κατά την διάρκεια της κωδικοποίησης θεωρούμε έναν σταθερό τύπο παραθύρου, στην περίπτωσή μας το “KBD”. Αφού διαβαστεί το αρχείο, το χωρίζουμε σε επικαλυπτόμενα frames των 2048 δειγμάτων, με επικάλυψη 50%, τα τροφοδοτούμε στην συνάρτηση SSC για να βρούμε τους τύπους τους, και αποθηκεύουμε τα αποτελέσματα στη μεταβλητή frameTypes. Απαιτείται ξεχωριστή προσέγγιση για την εύρεση των τύπων των οριακών frames. Για το πρώτο απλά αρχικοποιούμε τον τύπο του ως “OLS”. Οποιοδήποτε τύπο και να επιλέγαμε δεν έχει σημασία γιατί η εξάρτηση από τον τύπο του πρώτου frame εξαλείφεται. Για το τελευταίο, διαβάζουμε την τιμή του δεύτερου από το τέλος και σύμφωνα με αυτόν, το κατηγοριοποιούμε.

Στη συνέχεια, εφαρμόζουμε την filterbank σε όλα τα frames και αποθηκεύουμε για το κάθε ένα, τον τύπο του, τον τύπο του παραθύρου που χρησιμοποιήθηκε καθώς και τους συντελεστές MDCT για κάθε κανάλι. Όλα αυτά αποθηκεύονται στο struct AACSeq1.

2.5 *function* x = iAACoder1(AACSeq1, fNameOut)

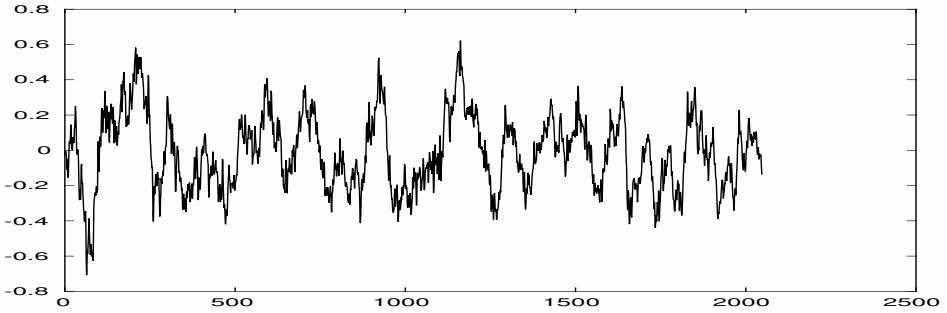
Αποτελεί την αντίστροφη συνάρτηση της AACoder1, υλοποιεί δηλαδή έναν αποκωδικοποιητή. Δέχεται ως όρισμα, τον πίνακα με structs που επιστρέφεται από τον AACoder1 και το όνομα που θα δοθεί στο αποκωδικοποιημένο αρχείο μαζί με το path του, ενώ η έξοδός του είναι ένας πίνακας που περιέχει τα δείγματα του αποκωδικοποιημένου αρχείου ήχου.

Κατά την αποκωδικοποίηση, τροφοδοτούνται οι συντελεστές MDCT των καναλιών στην iFilterbank, μαζί με τον τύπο του κάθε frame και τον τύπο του παραθύρου και αφού προστεθούν όλα τα frames, επιστρέφεται το αρχείο ολόκληρο στο πεδίο του χρόνου και αποθηκεύεται.

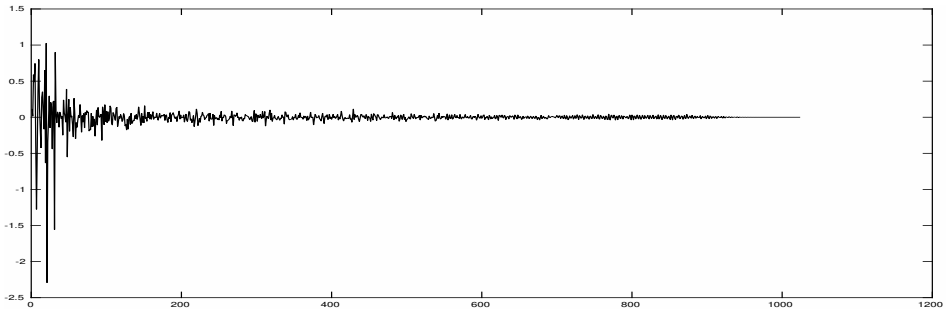
2.6 *function* SNR = demoAAC1(fNameIn, fNameOut)

Στην συνάρτηση αυτή, επιδεικνύεται η λειτουργία του κωδικοποιητή και του αποκωδικοποιητή και επιστρέφεται ο σηματοθορυβικός λόγος μεταξύ του αποκωδικοποιημένου και του αρχικού αρχείου ήχου. Δέχεται ως ορίσματα το path του αρχείου που επιθυμούμε να κωδικοποιήσουμε και το όνομα που θα του δοθεί μετά την αποκωδικοποίηση. Για την εύρεση του σηματοθορυβικού λόγου χρησιμοποιείται η συνάρτηση του MATLAB snr.

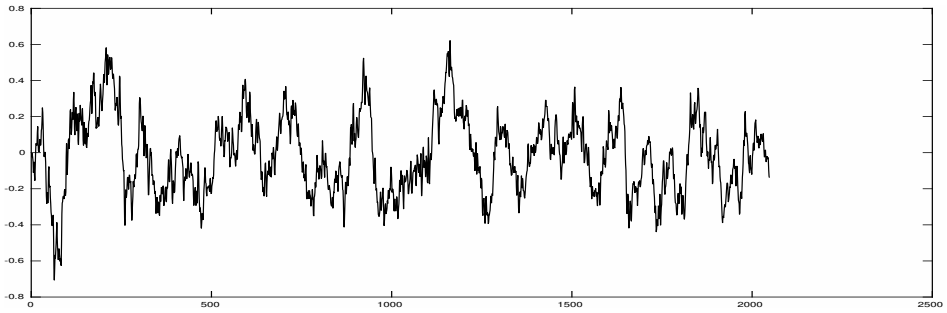
Παρακάτω φαίνεται η λειτουργία των παραπάνω συναρτήσεων στο 2ο frame του δείγματος ήχου “LicorDeCalandraca.wav” που δίνεται. Ο λόγος του SNR είναι 301.6522 dB χρησιμοποιώντας παράθυρα τύπου “KBD”.



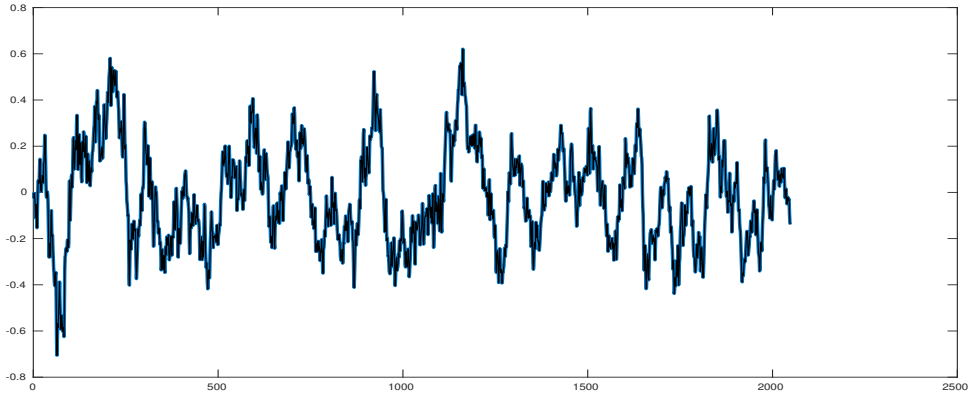
Σχήμα 1: Το frame στο πεδίο του χρόνου πριν τροφοδοτηθεί σε συναρτήσεις



Σχήμα 2: Μετάβαση στο πεδίο της συχνότητας με την συνάρτηση filterbank, συντελεστές MDCT



Σχήμα 3: Η επιστροφή στο πεδίου του χρόνου μέσω της συνάρτησης iFilterbank



Σχήμα 4: Κοινό διάγραμμα αρχικού σήματος με το σήμα εξόδου της iFilterbank

Όπως φαίνεται στο 4, το αρχικό σήμα με το σήμα εξόδου της iFilterbank ταυτίζονται. Η γραμμή

του ενός είναι συνειδητά πιο έντονη έτσι ώστε να είναι φανερή η ταύτιση. Αυτό είναι ένδειξη ότι οι υλοποιημένες συναρτήσεις εκτελούνται σωστά αφού ουσιαστικά δεν έχει χαθεί πληροφορία και η επανακατασκευή πρέπει να είναι σχεδόν τέλεια.

3 2ο Επίπεδο

Στο 2ο επίπεδο, υλοποιείται η βαθμίδα Temporal Noise Shaping (TNS), ένας κωδικοποιητής και οι αντίστροφες συναρτήσεις τους.

3.1 *function* [frameFout, TNScoeffs] = TNS(frameFin, frameType)

Η συνάρτηση TNS υλοποιεί τη βαθμίδα Temporal Noise Shaping για ένα κανάλι. Δέχεται ως εισόδους τους κβαντισμένους συντελεστές MDCT των 2 καναλιών ενός frame και τον τύπο του ενώ επιστρέφει σαν έξοδο το frame στο πεδίο συχνότητας, σε όρους MDCT συντελεστών έπειτα από την εφαρμογή του TNS καθώς και τους κβαντισμένους συντελεστές του TNS.

Σαν πρώτο βήμα, αρχικοποιούμε τον πίνακα των μπαντών του ψυχοακουστικού μοντέλου όπως δίνεται στο πρότυπο. Στη συνέχεια, κατατάσσουμε του συντελεστές MDCT στις μπάντες και βρίσκουμε την ενέργεια κάθε μπάντας. Έπειτα, υπολογίζουμε τους συντελεστές κανονικοποίησης, τους εξομαλύνουμε όπως υποδεικνύεται και κανονικοποιούμε τους συντελεστές MDCT ανάλογα με την ενέργεια της μπάντας στην οποία ανήκουν. Με την βοήθεια της συνάρτησης lpc του MATLAB, βρίσκουμε τους συντελεστές γραμμικής πρόβλεψης για κάθε frame (ή subframe αν ο τύπος του είναι "ESH") και τους κβαντίζουμε με τη βοήθεια της συνάρτησης quantiz για ομοιόμορφο κβαντιστή βήματος 0.1. Εφαρμόζουμε το FIR φίλτρο $H_{TNS}(z) = 1 - a_1 z^{-1} - a_2 z^2 - a_3 z^{-3} - a_4 z^{-4}$ στους αρχικούς συντελεστές MDCT όπου a_i οι κβαντισμένοι συντελεστές γραμμικής πρόβλεψης.

3.2 *function* frameFout = iTNS(frameFin, frameType, TNScoeffs)

Αποτελεί την αντίστροφη συνάρτηση της TNS. Δέχεται ως όρισμα το frame στο πεδίο του χρόνου σε όρους συντελεστών MDCT μετά το TNS, τον τύπο του και τους συντελεστές TNS, ενώ ως έξοδο επιστρέφει τους συντελεστές MDCT πριν το TNS. Για την εύρεση τους χρησιμοποιεί το αντίστροφο φίλτρο $H_{TNS}(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^2 - a_3 z^{-3} - a_4 z^{-4}}$

3.3 *function* AACSeq2 = AACoder2(fNameIn)

Υλοποιεί έναν κωδικοποιητή ο οποίος ουσιαστικά εκτελεί την συνάρτηση TNS. Δέχεται ως όρισμα το path του δείγματος ήχου που επιθυμούμε να κωδικοποίησουμε και επιστρέφει ένα struct μεγέθους K x 1, όπου K το πλήθος των frames που έχουν κωδικοποιηθεί και παριέχει για το κάθε ένα τον τύπο του, τον τύπο παραθύρου που χρησιμοποιήθηκε κατά την κωδικοποίηση, τους συντελεστές MDCT και τους κβαντισμένους συντελεστές TNS για κάθε κανάλι.

Όπως και στον παραπάνω κωδικοποιητή, αρχικά διαβάζεται το αρχείο, χωρίζεται σε frames και με την βοήθεια της συνάρτησης SSC, καθορίζεται ο τύπος τους με την ίδια αντιμετώπιση για τα οριακά frames. Στη συνέχεια, τροφοδοτούνται τα frames ως είσοδοι στην συνάρτηση filterbank και στην συνάρτηση TNS και αποθηκεύονται τα αποτελέσματα στο struct.

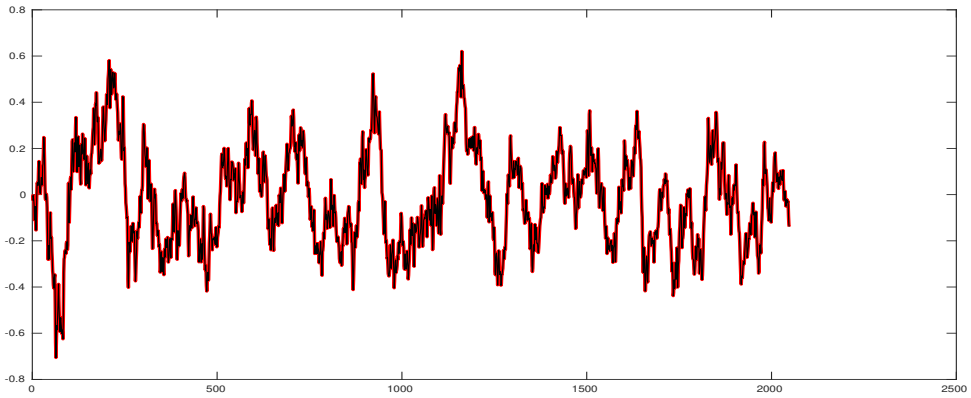
3.4 *function* x = iAACoder2(AACSeq2, fNameOut)

Υλοποιεί την αντίστροφη του κωδικοποιητή AACoder1, δηλαδή έναν αποκωδικοποιητή. Δέχεται ως εισόδους το K x 1 struct του κωδικοποιημένου αρχείου καθώς και το όνομα που πρόκειται να πάρει μετά την αποκωδικοποίησή του. Ως έξοδο, επιστρέφει τα αποκωδικοποιημένα δείγματα του αρχείου και τα αποθηκεύει σε μορφή WAV. Για να το επιτύχει αυτό, χρησιμοποιεί τις συναρτήσεις iTNS και iFilterbank.

3.5 *function* SNR = demoAAC2(fNameIn, fNameOut)

Η συνάρτηση αυτή επιδεικνύει την κωδικοποίηση του 2ου επιπέδου. Δέχεται ως ορίσματα το path του αρχείου ήχου που πρίκειται να κωδικοποιηθεί και το όνομα που πρόκειται να πάρει μετά την αποκωδικοποίησή του ενώ επιστρέφει τον συνολικό σηματοθορυβικό λόγο σε dB. Χρησιμοποιεί τις συναρτήσεις AACoder2 και iAACoder2 για την κωδικοποίηση και αποκωδικοποίηση του αρχείου ενώ για την εύρεση του σηματοθορυβικού λόγου χρησιμοποιεί την συνάρτηση snr του MATLAB.

Παρακάτω βλέπουμε τα αποτελέσματα του κωδικοποιητή και αποκωδικοποιητή.



Σχήμα 5: Κοινό διάγραμμα αρχικού σήματος με το σήμα εξόδου της iAACoder2

Να σημειωθεί ότι ο σηματοθορυβικός λόγος ισούται με 301.5897 dB που σημαίνει ότι δεν έχει αλλάξει σχεδόν καθόλου, γεγονός που είναι λόγικο αφού μέχρι στιγμής έχουμε κβαντίσει μόνο τους συντελεστές γραμμικής πρόβλεψης και όχι το σήμα άρα δεν έχουμε χάσει πληροφορία. Οι μικρή απόκλιση ανάμεσα στους δύο σηματοθορυβικούς λόγος έγκειται στις μικροαποκλίσεις των πράξεων λόγω των στρογγυλοποιήσεων που εκτελεί το MATLAB. Στο 5 είναι φανερή η ταύτιση του αρχικού σήματος με το σήμα εξόδου του αποκωδικοποιητή, ένδειξη ότι υλοποιήθηκε σωστά. Το ένα σήμα σχεδιάστηκε με πιο έντονη γραμμή και πάλι προς σκοπό οπτικοποίησης των αποτελεσμάτων.

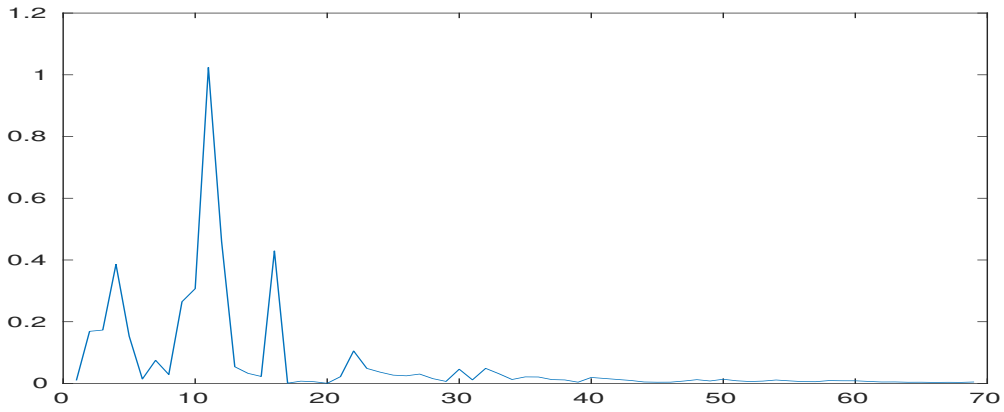
4 3ο Επίπεδο

Στο 3ο επίπεδο υλοποιείται το ψυχοακουστικό μοντέλο, οι βαθμίδες του κβαντιστή και του Huffman ένας κωδικοποιητής και οι αντίστροφές τους συναρτήσεις.

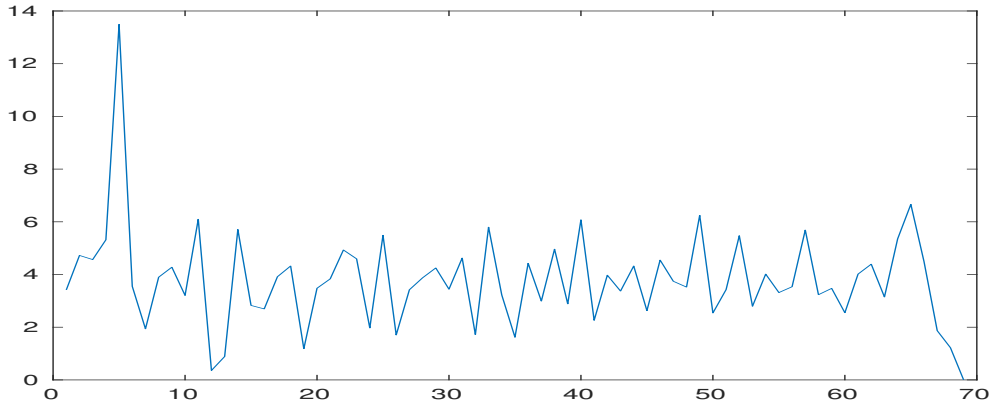
4.1 *function* SMR = psycho(frameT, frameType, frameTprev1, frameTprev2)

Υλοποιεί την βαθμίδα του ψυχοακουστικού μοντέλου για ένα κανάλι. Δέχεται ως είσοδο το frame στο πεδίο του χρόνου, τον τύπο του και τα 2 προηγούμενα του frames στο ίδιο κανάλι. Σαν έξοδο επιστρέφει τον λόγο σήματος προς μάσκα (Signal to Mask Ratio - SMR).

Ακολουθώντας την διαδικασία που περιγράφεται στο πρότυπο, βρίσκουμε το κατώφλι ακουστότητας που φαίνεται στο 13 ενώ στο 7 φαίνεται το SMR.



Σχήμα 6: Κατώφλι ακουστότητας για το αριστερό κανάλι του 2ου frame

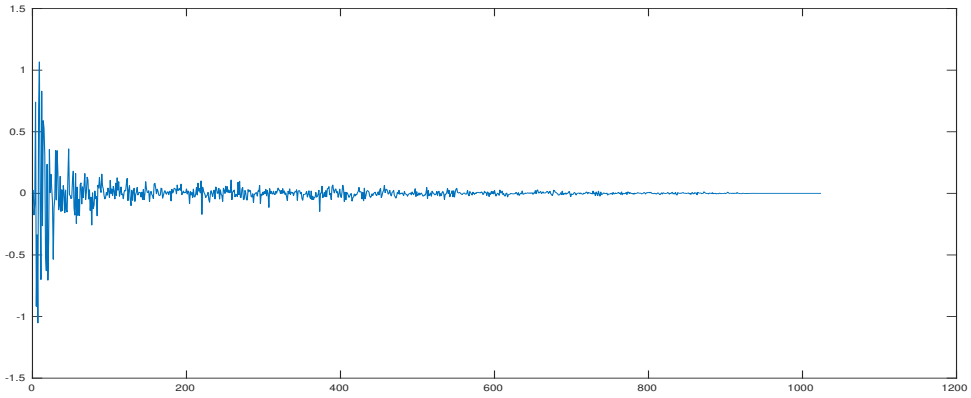


Σχήμα 7: Signal to Mask Ratio για το αριστερό κανάλι του 2ου frame

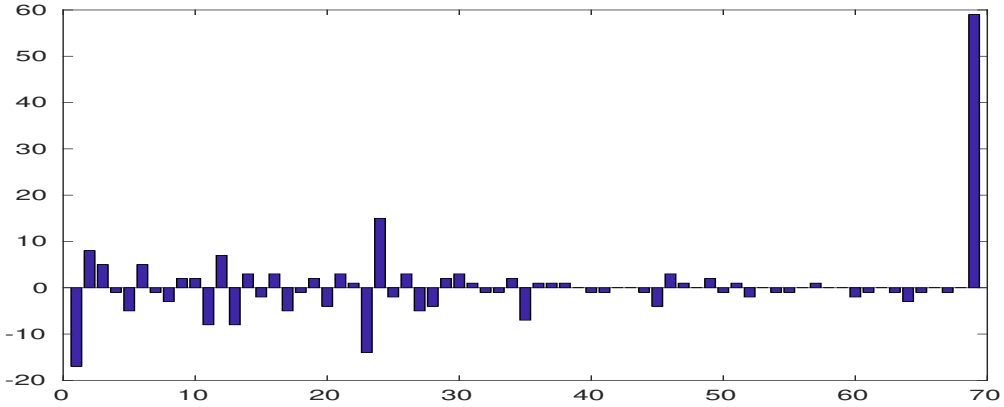
4.2 *function* [S, sfc, G] = AACquantizer(frameF, frameType, SMR)

Υπολογίζει εσωτερικά το κατώφλι ακουστότητας T(b) και υλοποιεί τη βαθμίδα του κβαντιστή για κάθε κανάλι. Δέχεται ως εισόδους το frame στο πεδίο της συχνότητας σε όρους συντελεστών MDCT, τον τύπο του και το υπολογισμένο SMR. Σαν έξοδο επιστρέφει τον πίνακα s που περιέχει τα σύμβολα κβάντισης των συντελεστών MDCT του frame, τον πίνακα sfc με τις τιμές των Scalefactor για κάθε Scalefactor band και το G που είναι το global gain του frame.

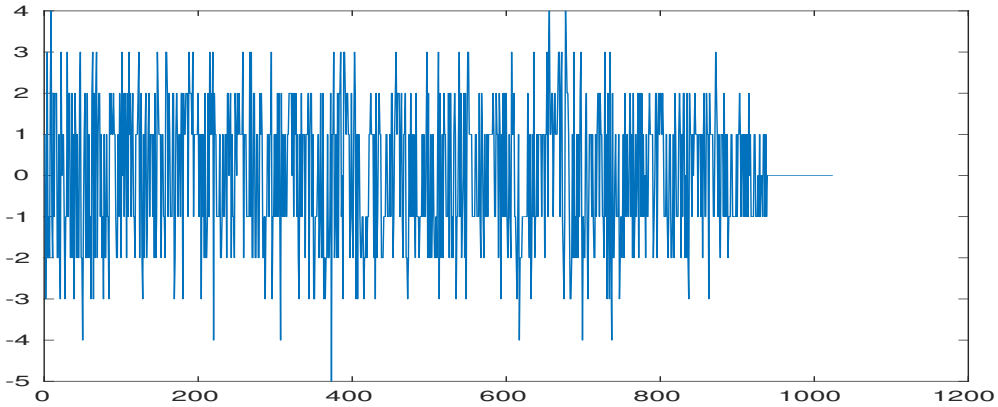
Η συνάρτηση παίρνει ως είσοδο τους συντελεστές MDCT που φαίνονται στο 8



Σχήμα 8: Συντελεστές MDCT πριν τον κβαντισμό



Σχήμα 9: Επίπεδα κβαντισμού



Σχήμα 10: Κβαντισμένοι συντελεστές

4.3 *function* frameF = iAACquantizer(S, sfc, G, frameType)

Υλοποιεί την αντίστροφη συνάρτηση της AACquantizer. Δέχεται ως ορίσματα τα σύμβολα κβάντισης των συντελεστών MDCT, τον πίνακα με τα scalefactors κάθε μπάντας και το global gain. Σαν

έξοδο, με τους συντελεστές και το scalefactor προσπαθεί να επανακατασκαευάσει το frame στο πεδίο της συχνότητας σε όρους συντελεστών MDCT, το οποίο και επιστρέφει.

4.4 *function* [huffSec, huffCodebook] = encodeHuff(coeffSec, huffLUT, forcedCodebook)

Υλοποιεί τη βαθμίδα κωδικοποίσης Huffman. Δέχεται ως εισόδους κβαντισμένους συντελεστές, τους Huffman look-up tables και τον δείκτη Codebook, όπως αυτός παρουσιάζεται στο πρότυπο. Σαν έξοδο, επιστρέφει ένα string από '1' και '0' αντίστοιχο με την κωδικοποίηση Huffman, και το huffCodebook, τον αριθμό των Huffman codebook που χρησιμοποιήθηκαν.

Για την υλοποίησή της ακολουθήθηκαν οι οδηγίες από το πρότυπο ενώ για να χρησιμοποιηθούν οι look-up tables κλήθηκε η συνάρτηση loadLUT.m.

4.5 *function* decCoeffs = decodeHuff(huffSec, huffCodebook, huffLUT)

Υλοποιεί την αντίστροφη συνάρτηση της encodeHuff, δηλαδή αποτελεί έναν αποκωδικοποιητή Huffman. Δέχεται ως είσοδο το string με '1' και '0' που αντιστοιχεί στην κωδικοποίηση Huffman, τους δείκτες codebook που χρησιμοποιήθηκαν και τους Huffman look-up tables. Σαν έξοδο, επιστρέφει τους αποκωδικοποιημένους κβαντισμένους συντελεστές.

Η διαδικασία είναι και πάλι αυτή που παρουσιάζεται στο πρότυπο ενώ λόγο του zero padding που είναι απαραίτητο, το μήκος της εξόδου, δηλαδή των κβαντισμένων συντελεστών, ίσως είναι μεγαλύτερο από αυτό που θα έπρεπε. Σε αυτήν την περίπτωση οι τιμές που είναι εκτός του εύρους του αναμενόμενου μήκους, θα πρέπει να αγνοηθούν ενώ αναμένεται να είναι και μηδενικοί

4.6 *function* AACSeq3 = AACoder3(fNameIn, fnameAACoded)

Υλοποιεί έναν κωδικοποιητή ο οποίος δέχεται σαν είσοδο το path του αρχείου το οποίο πρόκειται να κωδικοποιηθεί και το path του .mat αρχείου που πρόκειται να εγγραφεί μετά την κωδικοποίηση. Ως έξοδο, επιστρέφει ένα struct με διάσταση k x 1, όπου k, ο αριθμός των frames του αρχείου, το οποίο περιέχει για κάθε frame, όπως και πριν τον τύπο του και το παράθυρο που χρησιμοποιήθηκε,ενώ για κάθε κανάλι επιστρέφει τους κβαντισμένους συντελεστές TNS, τα κατώφλια ακουστότητας λόγω του ψυχοακουστικού μοντέλου, τα κβαντισμένα global gains, τους κωδικοποιημένους με Huffman κβαντισμένους συντελεστές MDCT και scalefactors και το Huffman codebook.

Φορτώνει αρχικά τους πίνακες TNS, διαβάζει το αρχείο προς κωδικοποίηση και αφού το χωρίσει σε frames χρησιμοποεί στη συνέχεια τις συναρτήσεις SSC, filterbank, TNS, psycho, AACquantizer και encodeHuff για να το κωδικοποίησει. Τα αποτελέσματα φαίνονται παρακάτω:

4.7 *function* x = iAACoder3(AACSeq3, fNameOut)

Υλοποιεί τον αντίστοιχο αποκωδικοποιητή. Δέχεται ως εισόδους, το struct που προκύπτει από τον κωδικοποιητή AACoder3 και το όνομα μαζί με το path που πρόκειται να πάρει το αποκωδικοποιημένο αρχείο ενώ ως έξοδο επιστρέφει έναν πίνακα με το αποκωδικοποιημένο σήμα. Για την υλοποίηση του χρησιμοποιούνται οι συναρτήσεις decodeHuff, iAACquantizer, iTNS και iFilterbank.

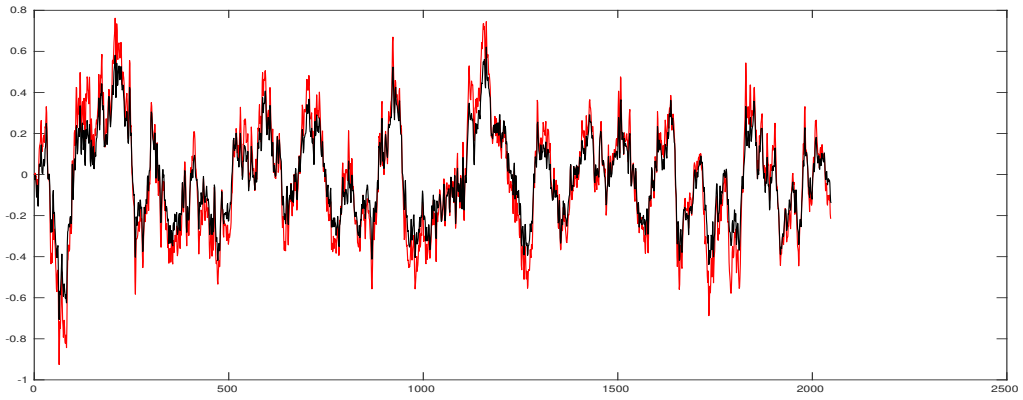
4.8 *function* [SNR, bitrate, compression] = demoAAC3(fNameIn, fNameOut, frameAACoded)

Ουσιαστικά η συνάρτηση αυτή παρουσιάζει την λειτουργία των συναρτήσεων του επιπέδου 3. Δέχεται ως εισόδους το path του αρχείου που πρόκειται να κωδικοποιηθεί και το όνομα του αρχείου που πρόκειται να επιστραφεί μετά την αποκωδικοποίηση. Σαν έξοδο, επιστρέφει τον σηματοθορυβικό λόγο του σήματος μετά την κωδικοποίηση και αποκωδικοποίησή του, το bitrate και το ποσοστό συμπίεσης.

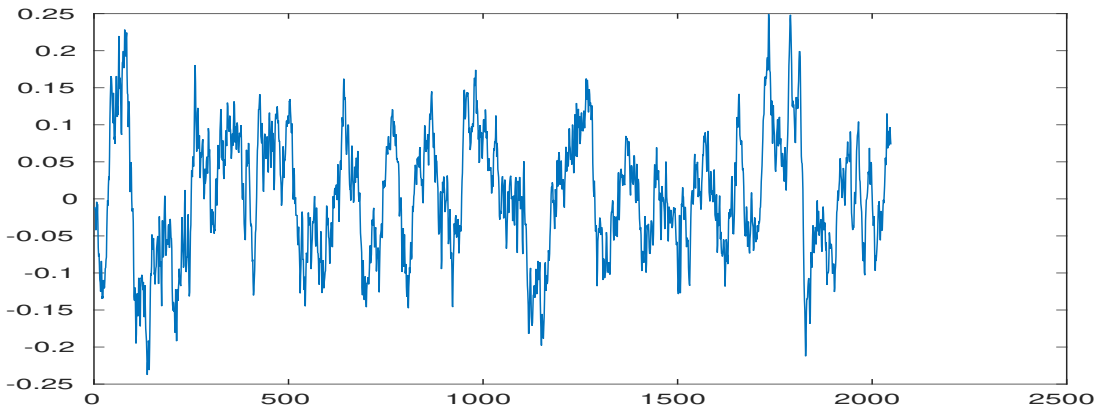
Αφού εκτελέστηκε η συνάρτηση, επέστρεψε τα εξής αποτελέσματα:

SNR	5.1821 dB
bitrate	3.1937e+05
compression	4.8094

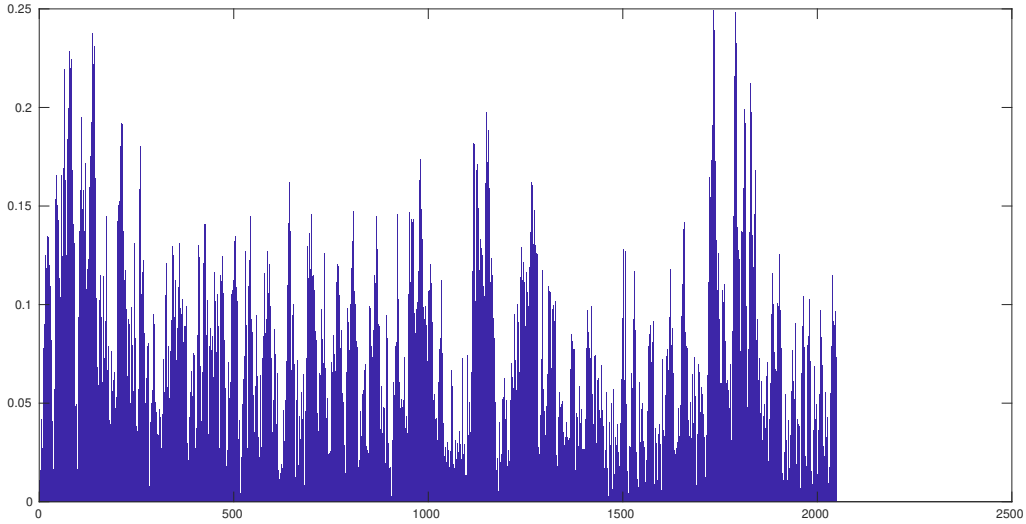
Πιο συγκεκριμένα, είδαμε τα αποτελέσματα του κωδικοποιητή και αποκωδικοποιητή για το 2ο frame του δείγματος ήχου:



Σχήμα 11: Αρχικό (μαύρο) και αποκωδικοποιημένο (κόκκινο) σήμα



Σχήμα 12: Διαφορά σημάτων σε κάθε δείγμα



Σχήμα 13: Απόλυτη διαφορά σημάτων

Όπως βλέπουμε το αποκωδικοποιημένο και το αρχικό σήμα δεν ταυτίζονται. Αυτό είναι απολύτως λογικό αφού το σήμα πλέον έχει αλλοιωθεί και δεν ήταν δυνατό να επανακατασκευαστεί πλήρως. Για τον λόγο αυτό είναι και ο σηματοθορυβικός λόγος πολύ μικρός πλέον σε σχέση με τους προηγούμενους.