Συστήματα Πολυμέσων

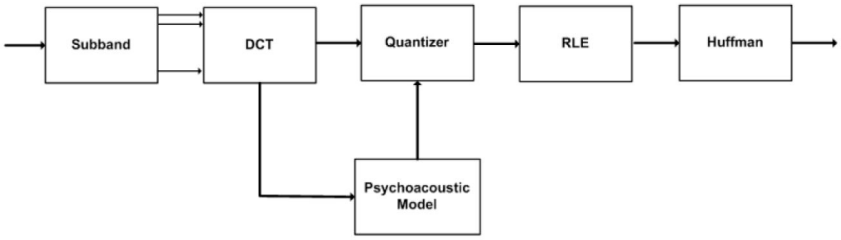
Απλοποιημένος codec MP3

Κουτρουμπής Γεώργιος, ΑΕΜ: 9668
Κυργιαφίνη Αγγέλη Δήμητρα, ΑΕΜ: 9685

2023

Περιεχόμενα

Σκοπός της εργασίας αυτής είναι η υλοποίηση ενός απλοποιημένο MP3 codec. Τα βήματα φαίνονται στο παρακάτω διάγραμμα. Η αναφορά αυτή περιέχει επεξήγηση του κωδικά και συμπεράσματα που εξήχθησαν.



Εικόνα 1. *Μπλοκ διάγραμμα της διαδικασίας*

1 Subband Filtering

Για αυτήν την βαθμίδα υλοποιούνται οι συναρτήσεις coder, decoder, codec, οι οποίες υλοποιούν την ανάλυση του αρχείου ήχου σε frames και έπειτα τη σύνθεση του ανακασκευασμένου αρχείου ήχου από τα frames, χωρίς κάποια επεξεργασία των frames (στο αρχείο subband_filtering0.py). Οι αντίστοιχες συναρτήσεις που επιτελούν το σύνολο της διαδικασίας βρίσκονται στο αρχείο simple_mp3_codec.py.

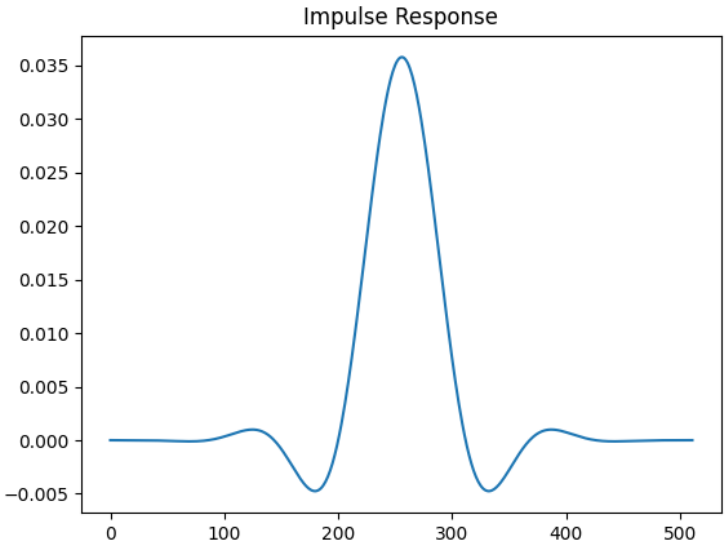
1.1 Coder

Αρχικά, βρίσκουμαι τον αριθμό των frames που χρειάζονται για το αρχείου ήχου. Θεωρώντας ότι κάθε frame έχει $N * M$ στοιχεία, χρειάζονται frames ίσα με το ταβάνι της διαίρεσης $sound\_file\_length/(N * M)$, δηλαδή

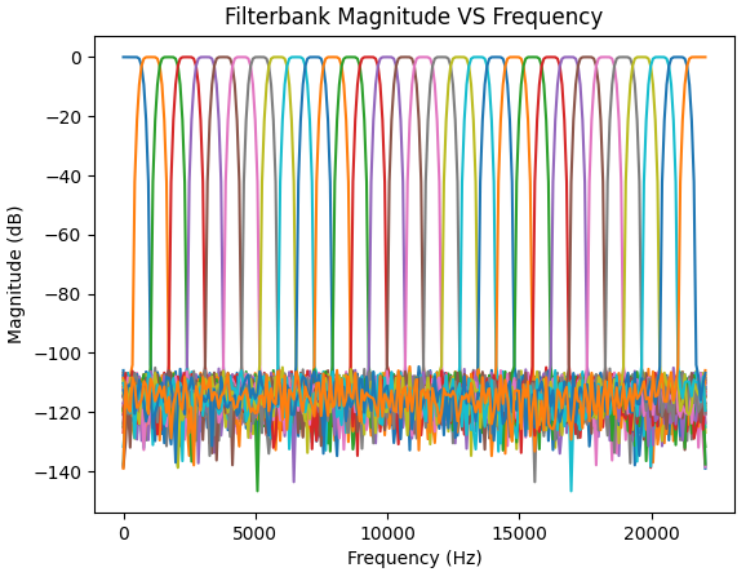
$$ceil(sound\_file\_length/(N * M))$$

frames, όπου $sound\_file\_length$ ο αριθμός των δειγμάτων στο αρχείο ήχου. Οπότε αρχικά προστίθονται μηδενικά στο αρχείο έτσι ώστε να εκτελείται ακριβώς η παραπάνω διαίρεση, δηλαδή να είναι το μέγεθος του ενός frame ακέραιο πολλαπλάσιο του αριθμού δειγμάτων του αρχείου ήχου.
Το frame χωρίζεται σε μπάντες συχνότητας, οπως φαίνεται στην παρακάτω εικόνα. Το σήμα πολλαπλασιάζεται με M=32 διαφορετικά παράθυρα και ο κάθε ένας από τους πολλαπλασιασμούς μας δίνει μία μπάντα. Έπειτα τα δείγματα κάθε μπάντας υποδειγματοληπτούνται και προκύπτει ένας δισδιάστατος πίνακαςπου για κάθε μπάντα περιέχει N=36 δείγματα.
Επειδή όμως για τη δημιουργία κάθε frame, απαιτούνται L-M (L μήκος κρουστικής απόκρισης) δείγματα από την επόμενη επανάληψη, προστίθονται και L-M μηδενικά στο τελός, για την τελευταία επανάληψη/τελευταίο frame.

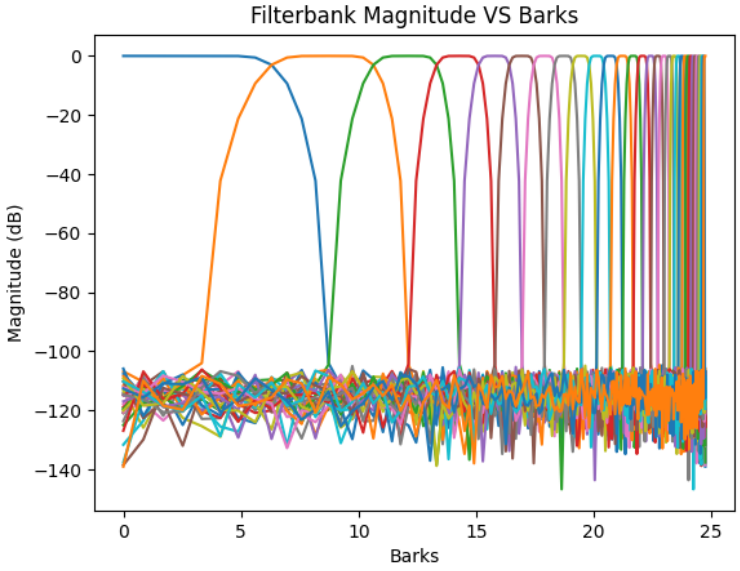
Οπότε, σε κάθε επανάληψη ο buffer έχει μήκος (N-1)*M + L. Από τον διαχωρισμό σε μπάντες και την υποδειγματοληψία κάθε frame (μέσω της frame sub analysis) προκύπτει ένας NxM πίνακας ο οποίος περιέχει τα N στοιχεία για τις M μπάντες του frame. Σε αυτό τον πίνακα που προκύπτει για κάθε frame εκτελούνται και οι υπόλοιπες βαθμίδες της κωδικοποίησης, οι οποίες εξηγούνται σε επόμενες ενότητες. Η συνάρτηση coder επιστρέφει τελικά τα παραγώμενα frames.
Η τελική υλοποίηση της συνάρτησης coder, η MP3cod, εκτελεί τα παραπάνω, αλλά επιπλέον, επεξεργάζεται κάθε frame, σύμφωνα με την διαδικασία που ζητείται. Τελικά, δημιουργεί ένα αρχείο με την παραγώμενη κωδικοποίηση Huffman, καθώς και επιστρέφει τις πληροφορίες που χρειάζονται για την αποκωδικοποίηση κάθε frame.



Εικόνα 2. *Κρουστική απόκριση πρότυπου βαθυπερατού φίλτρου*



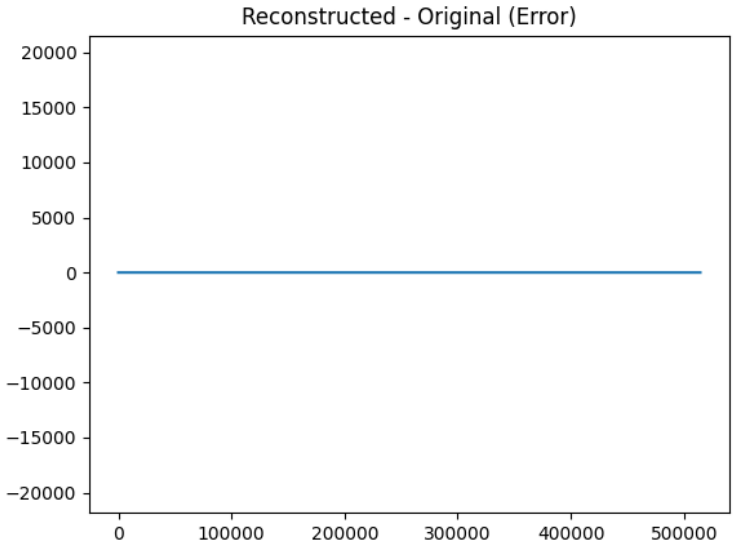
Εικόνα 3. *Μέτρο συναρτήσεων μεταφοράς ως προς τη συχνότητα f (Hz)*



Εικόνα 4. *Μέτρο συναρτήσεων μεταφοράς ως προς τη συχνότητα z (barks)*

1.2 Decoder

Η συνάρτηση decoder λαμβάνει ως είσοδο την έξοδο του coder, και με παρόμοιο τρόπο συνθέτει ένα ανακατασκευασμένο αρχείο ήχου. Για κάθε frame, ο buffer χρησιμοποιεί και τις L/M πρώτες γραμμές του επόμενου frame για τη σύνθεση των ανακατασκευασμένων δειγμάτων. Για αυτό το λόγω, κατά αντιστοιχία με πάνω, προστίθονται L/M σειρές μηδενικών, στον πίνακα που περιέχει τα frames vertically stacked.
Η τελική υλοποίηση του decoder, η MP3decod, εκτελεί τα παραπάνω, αλλά επιπλέον, για να παράξει τα αρχικά frames, εκτελεί τις αντίστροφες διαδικασίες από την MP3cod. Τελικά, παράγεται ένα αποκωδικοποιημένο αρχείο ήχου.



Εικόνα 5. *Η διαφορά μεταξύ ανακατασκευασμένου και αρχικού αρχείου, χωρίς επεξεργασία των frames*

Όπως φαίνεται, με το διαχωρισμό του αρχείου σε frames με subband filtering, και μετά την ανακατσκευή του, χωρίς επεξεργασία των frames, επιτυγχάνεται σχεδόν τέλεια ανατασκευή (υπάρχουν σφάλματα μεγέθους 1-2). Λόγω, όμως, του πως λειτουργεί η διαδικασία filterbanks reconstruction, γίνεται μετατόπιση του αρχικού σήματος κατά L-M δείγματα. Αυτό σημαίνει ότι στο ανακατασκευασμένο σήμα, το σήμα ξεκινάει από το L-M δείγμα του αρχικού αρχείου, ενώ τα τελευταία L-M δείγματα είναι μηδενικά. Συνεπώς έχει ίδιο αριθμό δειγμάτων με το αρχικό, αλλά είναι μετατοπισμένο κατά L-M αριστερά. Ακουστικά τα δύο αρχεία ακούγονται ακριβώς ίδια. Οπότε η επιρροή των "χαμένων" αρχικών δειγμάτων ή η μικρή διαφορά μεταξύ των τιμών μερικών δειγμάτων δεν γίνονται αντιληπτές.
Το SNR των δύο σημάτων είναι $SNR = 72.32$. Το SNR υπολογίζεται με τον

εξής τύπο:

$$SNR = 10 * log_{10}(\frac{mean(original\_shifted^2)}{mean(reconstructed\_shifted^2)})$$

όπου με επίθεμα shifted εννούμε ότι τα δύο σήματα έχουν μετατοπιστεί έτσι ώστε τα δείγματά τους να κάνουν align.
Οι συναρτήσεις codec και MP3codec απλώς καλούν τις coder, decoder και MP3cod, MP3decod αντίστοιχα.

2 DCT

Αρχικά υπολογίζονται οι συντελεστές DCT ενός frame, εφαρμόζοντας τη συνάρτηση dct της βιβλιοθήκης scipy, με DCT τύπου 2, με ορθοκανονικοποίηση, για κάθε στήλη ενός frame ξεχωριστά, και συνένωση των συντελεστών σε ένα ενιαίο διάνυσμα.
Type II DCT:

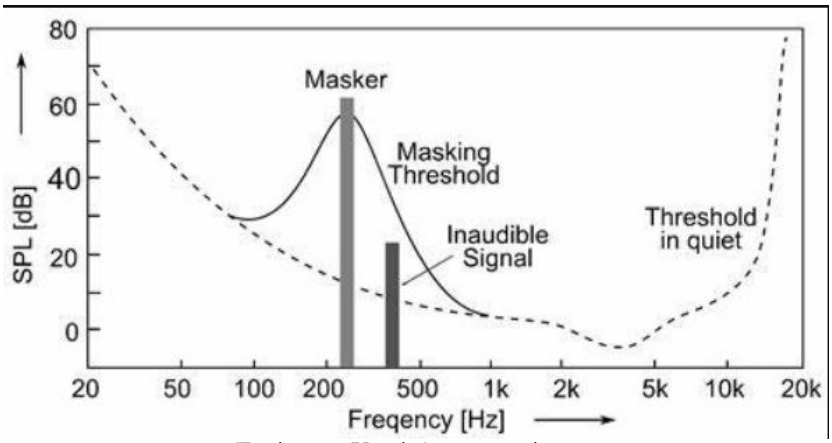
$$y_k = 2\sum_{n=0}^{N-1} x_n cos(\frac{\pi k(2n+1)}{2N})$$

πολλαπλασιασμένο το y_k με συντελεστή f:

$$f = \begin{cases} \sqrt{\frac{1}{4N}} & \text{αν k=0} \\ \sqrt{\frac{1}{2N}} & \text{αλλιώς} \end{cases}$$

3 Psychoacoustic Model

Σε αυτό το βήμα της κωδικοποίησης υπολογίζονται τα tonal components (maskers). Το ψυχοακουστκό φανόμενο συνοψίζεται στο ότι το ανθρώπινο αυτί δεν μπορεί να αντιληφθεί οποιαδήποτε ένταση ενός ηχητικού κύματος. Η ένταση θα πρέπει να υπρβαίνει ένα κατώφλι για να μπορεί να είναι ακουστή. Το κατώφλι αυτό εξαρτάται από την συχνότητα.
Όταν υπάρχει σιγή και δεν υπάρχει τόνο σε κάποια συχνότητα, το κατώφλι ακουστότητας διαμορφώνεται όπως φαίνεται η διακεκομμένη γραμμή στην παρακάτω εικόνα. Όταν σε μία συχνότητα υπάρχει τόνος (masker), τότε επηρεάζεται το κατώφλι ακουστότητας των γύρω συχνοτήτων και αυξάνεται. Σε αυτό το κομμάτι της εργασίας υπολογίζεται το κατώφλι ακουστότητας.



Εικόνα 6. *Κατώφλι ακουστότητας*

Αρχικά υπολογίζονται τα tonal components. Για τον υπολογισμό της ισχύς των συντελεστών DCT γίνεται χρήση της σχέσης:

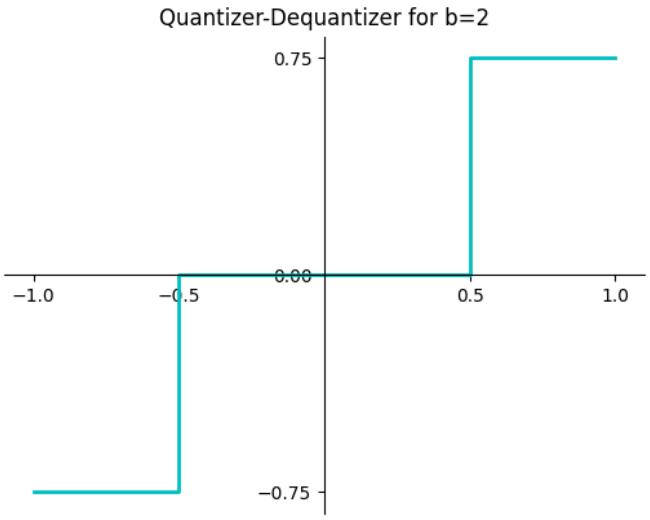
$$P(k) = 10log_{10}(|c(k)|^2)$$

. Έπειτα για μια αρχική θεώρηση των tonal components (maskers), συγκρίνεται αυτή η ισχύς με την ισχύ τόσο των άμεσα γειτονικών συχνοτήτων, όσο και μεταξύ των συχνοτήτων που βρίσκονται εντός μιας γειτονιάς Δ_k για την διακριτή συχνότητα k.
Πιο συγκεκριμένα ελέγχεται αν είναι μεγαλύτερος από τους κατά ένα γείτονές του ή αν είναι μεγαλύτερη η ισχύς του κατά τουλάχιστον 7 dB από τους Δ_k γείτονες από κάθε μεριά. Για την θεώρηση των Δ_k γειτόνων γίνεται χρήση της δοθείσας σχέσης, η οποία υλοποιείται στη συνάρτηση Dksparse.
Η Dksparse επιστρέφει έναν πίνακα διαστάσεων Kmax X Kmax (στην υλοποίησή μας Kmax = M*N = 1152). Κάθε σειρά k του πίνακα αντιστοιχεί σε μια διακριτή συχνότητα από 0 έως 1151, και κάθε στοιχείο (k,j) του πίνακα έχει τη τιμή 0 εάν δεν ανήκει στη γειτονιά του k, ή τη τιμή 1 αν ανήκει σε αυτήν. Στην υλοποίησή μας ο πίνακας που επιστρέφει η Dksparse περιέχει απευθείας την πληροφορία για τους γείτονες και δεξιά και **αριστερά** του k. Για παράδειγμα, αν k = 5, σύμφωνα με τον τύπο που δίνεται $\Delta\kappa = 2$. Οπότε ο πίνακας που επιστρέφει η Dksparse, στη σειρά 5, τα στοιχεία (5,7) και (5,3) θα έχουν τιμή 1, ενώ τα άλλα της σειράς τιμή 0.
Στη συνέχεια, μειώνεται τον αριθμό των maskers. Αρχικά, απορρίπτονται αυτοί που βρίσκονται κάτω από το κατώφλι ακουστότητας. Έπειτα, για κάθε ζεύγος maskers που απέχουν λιγότερο από 0.5 barks, αφαιρείται ο μικρότερος του ζεύγους. Αυτό γιατί το ανθρώπινο αυτί δεν είναι ευαίσθητο σε τόσο μικρές διαφορές, οπότε και πάλι η απώλεια αυτή της πληροφορίας δεν θα γίνει αντιληπτή από τον ακροατή.
Ο υπολογισμός που ακολουθεί αφορά το πως αυτοί οι τόνοι παραμορφώνουν το κατώφλι ακουστότητας της γύρω περιοχής. Συγκεκριμένα, υπολογίζεται πώς κάθε masker k επηρεάζει το κατώφλια κουστότητας κάθε συχνότητας i. Τελικά, το

global masking threshold υπολογίζεται για κάθε συχνότητα i συνδυάζοντας τα masking thresholds των maskers και του κατωφλιού στη σιγή.
Καθώς η υλοποίηση αποτελεί απλούστευση της πλήρης διαδικασίας για (απο)κωδικοποίηση MP3, η καμπύλη T_g που υπολογίζεται, πρέπει να μετατοπιστεί για να υπάρξει καλύτερο ακουστικό αποτέλεσμα. Η μετατόπιση αυτή σχολιάζεται αργότερα κατά την παράθεση των τελικών αποτελεσμάτων.

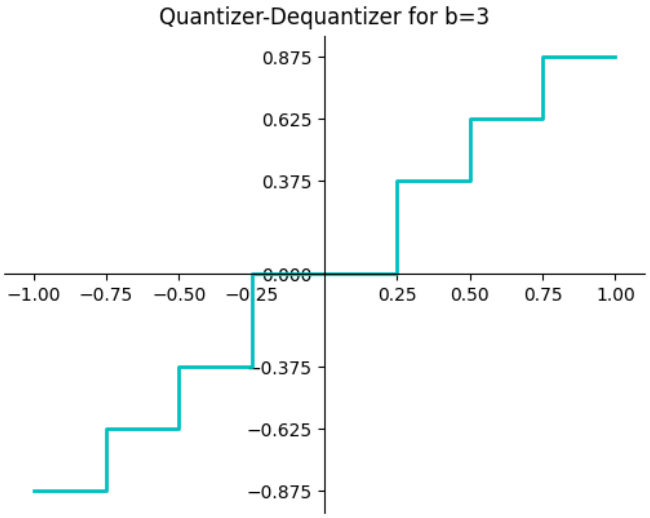
4 Κβαντιστής - Αποκβαντιστής

Ο κβαντιστής που ζητείται είναι ένας κβαντιστής ζώνης, όπου η νεκρή ζώνη έχει πλάτος $2w_b$, ενώ οι υπόλοιπες w_b, με $w_b = \frac{1}{2^{b-1}}$. Οπότε οι στάθμες κβαντισμού είναι τελικά $N = 2^b - 1$. Οι τιμές αποκβαντισμού βρίσκονται στο μέσο της κάθε στάθμης.
Παρακάτω φαίνονται γραφικά δύο παραδείγματα, κάνοντας χρήση 2 και 3 bits.
Για 2 bits το πλάτος $w_b = \frac{1}{2} = 0.5$, και οι στάθμες κβαντισμού $N = 2^2 - 1 = 4 - 1 = 3$.



Εικόνα 7. *Ο κβαντιστής-αποκβαντιστής για 2 bits*

Για 3 bits το πλάτος $w_b = \frac{1}{4} = 0.25$, και οι στάθμες κβαντισμού $N = 2^3 - 1 = 8 - 1 = 7$.



Εικόνα 8. *Ο κβαντιστής-αποκβαντιστής για 3 bits*

Οι τιμές αποκβαντισμού και για τα δύο παραδείγματα φαίνονται στις γραφικές παραστάσεις.

5 RLE

Για την κωδικοποίηση run-length, έγινε υλοποιήση following zeros run-length encoding. Αυτό σημαίνει ότι κωδικοποιείται κάθε μη-μηδενικός αριθμός (σύμβολο) σε ένα tuple που περιέχει τον αριθμό και τον αριθμό των μηδενικών αριστερά του (εώς το προηγούμενο μη-μηδενικό σύμβολο). Έχει δηλαδή τη μορφή $(symbol, following\_zeros)$. Τυχόν μηδενικά που υπάρχον στο τέλος της ακολουθίας, δεν κωδικοποιούνται, καθώς για την αντίστροφη διαδικασία, δίνεται ως όρισμα το μήκος της αρχικής ακολουθίας, οπότε αρκεί να αρχικοποιηθεί η απωδικοποιημένη ακολουθία σε μηδενικά.
Για παράδειγμα έστω η ακολουθία 00100003002046000. Με το rle θα παραχθούν τα εξής σύμβολα: (1,2) , (3,4), (2,2), (4,1), (6,0). Όπως φαίνεται, τα τελευταία τρία 0 δεν θα κωδικοποιηθούν. Για την αντίστροφη διαδικασία, αρχικα θα αρχικοποιηθεί μια ακολουθία μηδενικών, μήκους ίδιου με της αρχικής, δηλαδή στο παράδειγμα ίσο με 17. Στη συνέχεια για κάθε ζεύγος (symbol, following_zeros), θα προστίθεται το symbol στη θέση index+following_zeros, όπου η θέση index αποθηκεύει την θέση που πρέπει να προστεθεί ένα νέο σύμβολο, εάν δεν έχει μηδενικά αριστερά του (αρχικοποιείται στο 0. Zero-based). Οπότε στο παραπάνω παράδειγμα:

- Αρχικοποίηση σε μηδενικά (00000000000000000)

- Διαβάζεται το (1,2). Οπότε θα προστεθεί το σύμβολο 1 στη θέση $0+2=2$, και update το $index = 2+1 = 3$ (00100000000000000)
- Διαβάζεται το (3,4). Οπότε θα προστεθεί το σύμβολο 3 στη θέση $3+4=7$, και update το $index = 7+1 = 8$ (00100003000000000)
- Διαβάζεται το (2,2). Οπότε θα προστεθεί το σύμβολο 2 στη θέση $8+2=10$, και update το $index = 10+1 = 11$ (00100003002000000)
- Διαβάζεται το (4,1). Οπότε θα προστεθεί το σύμβολο 4 στη θέση $11+1 = 12$, και update το $index = 12+1 = 13$ (00100003002040000)
- Διαβάζεται το (6,0). Οπότε θα προστεθεί το σύμβολο 6 στη θέση $13+0 = 13$, και update το $index = 13+1 = 14$ (00100003002046000)
- Η διαδικασία ολοκληρώνει, καθώς δεν υπάρχουν άλλα ζεύγη να διαβαστούν και τελικά έχει αναπαραχθεί η αρχική ακολουθία

6 Κωδικοποίηση Huffman

Σε αυτό το στάδιο τα σύμβολα που παράχθηκαν από το run-length encoding κωδικοποιούνται κατά Huffman. Η κωδικοποίηση γίνεται κατά τα γνωστά για κάθε μοναδικό σύμβολο (symbol, following_zeros), παίρνοντας ως πιθανότητα το ποσοστό εμφάνισης του μοναδικού συμβόλου στο σύνολο των συμβόλων που παρήχθησαν από το rle.
Για την υλοποίηση της κωδικοποίησης Huffman, γίνεται χρήση κόμβων (nodes), που παρέχουν αναφορά στους 2 γειτονικούς (δεξί και αριστερό) γείτονες, το σύμβολο που περιέχουν (αν πρόκειται για φύλλο του δέντρου Huffman) και τον κώδικα τους (μια σειρά από bits).
Για την αποκωδικοποίηση του κώδικα Huffman ενός frame, παρέχονται οι πιθανότητες κάθε συμβόλου που βρέθηκαν κατά την κωδικοποίηση. Πάλι δημιουργείται το δέντρο Huffman. Στη συνέχεια, διαβάζονται bits από το bitstream που δίνεται για αποκωδικοποίηση, και "κινούμαστε" στο δέντρο αναλόγως, ξεκινώντας από τη ρίζα του δέντρου Huffman. Διαβάζονται bits μέχρι να φτάσουμε σε φύλλο, οπότε το σύμβολο που αντιστοιχεί στο σύνολο των bits που διαβάστηκαν από την τελευταία φορά που βρέθηκε σύμβολο (ή από την αρχή του bitstream, αν είναι το πρώτο σύμβολο) έως αυτό το βήμα, είναι το σύμβολο του φύλλου. Η διαδικασία συνεχίζεται εκ νέου, ξεκινώντας πάλι από την ρίζα, έως να διαβαστούν όλα τα bits.

7 Τελικά Αποτελέσματα

7.1 Demo αρχεία

Στην εργασία περιέχονται και μερικά demo αρχεία που επιδεικνύουν την ορθή λειτουργία του κώδικα.

- **demo1.py**: Demo που αφορά το πρώτο ερώτημα της εργασίας, δηλαδή εκτελεί την ανάλυση σε frames και μετά τη σύνθεση ενός αρχείου (που ορίζεται εντός του demo1 και μπορεί να αλλαχθεί στον κώδικα για χρήση οποιουδήποτε (μονοφονικού) αρχείου ήχου), χωρίς επεξεργασία των εκάστοτε frames. Παράγει ένα αρχείο wav με τον reconstructed ήχο (reconstructed_file0.wav) και τα σχετικά διαγράμματα για τα filterbanks, καθώς υπολογίζει και το SNR των 2 αρχείων.

- **demo2.py**: Demo που εκτελεί την συνολική διαδικασία του MP3 codec για ένα αρχείο ήχου (που ορίζεται εντός του demo1 και μπορεί να αλλαχθεί στον κώδικα για χρήση οποιουδήποτε (μονοφονικού) αρχείου ήχου), και παράγει ένα αρχείο ascii με την παραγώμενη κωδικοποίηση Huffman για το αρχείο καθώς (bitstream.txt) και ένα αρχείο wav με τον αποκωδικοποιημένο ήχο (xhat.wav), καθώς υπολογίζει και το SNR των 2 αρχείων.

- **demo_quantization.py**: Demo που παράγει ένα διάγραμμα για τον κβαντιστή/αποκβαντιστή που χρησιμοποιείται, για το δοσμένο αριθμό bits b.

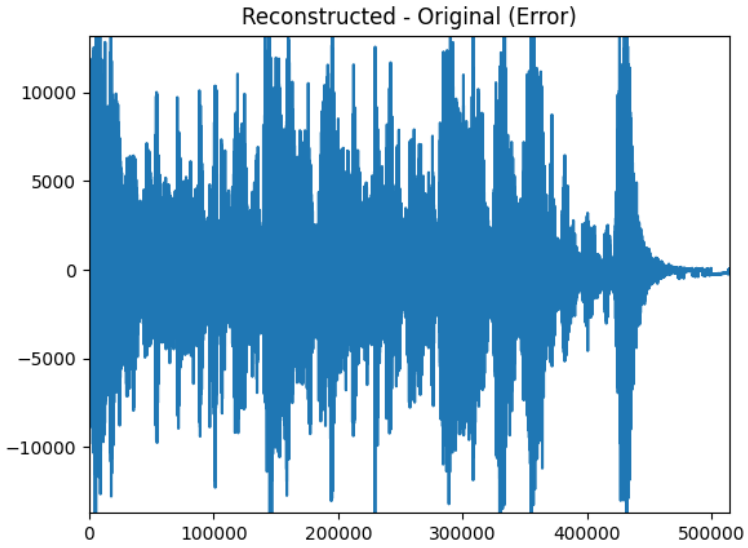
7.2 Σχόλια

Παρακάτω θα παρουσιαστούν τα αποτελέσματα της συνολικής διαδικασίας για το δοκιμαστικό αρχείο ήχου που δόθηκε. Όπως αναφέρθηκε, χρειάστηκε η μετατόπιση του κατωφλιού T_g, καθώς χωρίς αυτήν το αποκωδικοποιημένο αρχείου ήχου δεν ακουγόταν επαρκώς καλά. Για να βρεθεί μια επαρκής μετατόπιση, έγιναν δοκιμές για διαφορετικές μετατοπίσεις. Με μηδενική μετατόπιση, με μετατόπιση -5 και -10, το αποκωδικοποιημένο αρχείο απείχε πολύ ακουστικά από το αρχικό αρχείο, ακόμη και αν επιτυγχάνεται υψηλός βαθμός συμπίεσης.

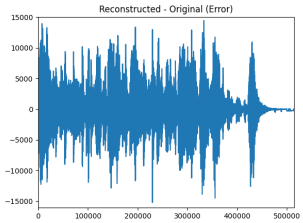
Μετατόπιση T_g	0	-5	-10	-15	-20
SNR	-0.2	2.77	4.37	8.92	12.58
File Size (kB)	69.71	125.86	212.35	344.74	528
Compression Ratio	14.42	7.99	4.73	2.91	1.9

Πίνακας 1. *Μετρήσεις για διάφορες μετατοπίσεις του κατωφλιού T_g*

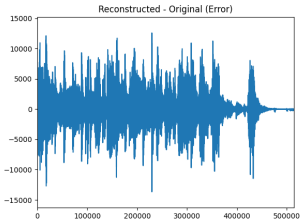
Οι μετατοπίσεις -15 και -20 παράγουν τα καλύτερα ακουστικά αποτελέσματα, ειδικότερα η μετατόπιση -20. Ενδεικτικά επιλέγεται η -20 ως η τελική μετατόπιση της εργασίας, καθώς παράγει αισθητά καλύτερο ακουστικό αποτέλεσμα από την -15, ενώ έχει βαθμό συμπίεσης ≈ 2.



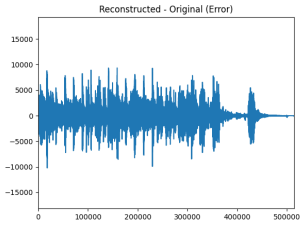
Εικόνα 9. *Σφάλμα μεταξύ αποκωδικοποιήμενου και αρχικού αρχείου, για μηδενική μετατόπιση του T_g*



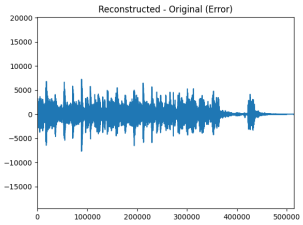
(a) *Μετατόπιση -5*



(b) *Μετατόπιση -10*



(c) *Μετατόπιση -15*



(d) *Μετατόπιση -20*

Εικόνα 10. *Σφάλμα μεταξύ αποκωδικοποιήμενου και αρχικού αρχείου, για διάφορες μετατοπίσεις του T_g*

Σημείωση: Για τον υπολογισμό του κωδικοποιημένου/συμπιεσμένου αρχείου μετρήθηκε το μέγεθος του αρχείου που περιέχει το bitstream που παράγεται από την κωδικοποίηση Huffman. Για την αποκωδικοποίηση των frames, απαιτούνται και παραπάνω πληροφορίες, όπως ο αριθμός των bits που χρησιμοποιήθηκαν για την κωδικοποίηση κάθε critical band, τα scaling factors, το μήκος του bitstream για το κάθε frame. Αυτά τα στοιχεία δίνονται στον MP3decod με τη μορφή μιας λίστας από dictionaries, έχοντας ένα dictionary για κάθε frame.