Multimedia Project

Fotiou Dimitrios
AEM 9650
Aristotle University of Thessaloniki
Department of Electrical and Computer Engineering

***Abstract*—**Το έγγραφο αυτό αποτελεί την εργασία στο μάθημα Συστήματα Πολυμέσων**.** Σκοπός της εργασίας είναι η υλοποίηση μιας απλουστευμένης εκδοχής του προτοκόλου **MP3.** Παρακάτω παρουσιάζονται τα διάφορα μέρη του προτοκόλου αυτού καθώς και οι λεπτομέρειες υλοποίησης του**.** Τέλος παρουσιάζονται και ορισμένα αποτελέσματα για την ποιότητα της κωδικοποίησης**.**

***Index Terms*—multimedia, MP3, codecs, audio**

I. Introduction

Το πρωτόκολλο MP3 [1] είναι μία μορφή συμπίεσης ήχου που χρησιμοποιείται για την αποθήκευση και μεταφορά αρχείων ήχου στο Διαδίκτυο και σε συσκευές αναπαραγωγής ήχου. Το MP3 αποτελείται από ένα αλγόριθμο συμπίεσης που μειώνει την ανάλυση του ήχου, χωρίς να επηρεάζει σημαντικά την ποιότητα του. Το MP3 είναι ένα από τα πιο διαδεδομένα πρωτόκολλα συμπίεσης ήχου και βασίζεται στο πρωτόκολλο ISO/IEC 11172-3.

Τα βασικά κομμάτια του MP3 είναι μια συχνοτική ανάλυση του σήματος εισόδου και ασυσχέτιστη των δειγμάτων. Έπειτα μια ελεγχόμενη παραμόρφωση του σήματος που επηρεάζεται από μια ψυχοακουστική ανάλυση που μειώνει την ακρίβεια κβαντισμού σε περιοχές του φάσματος που το σφάλμα κβαντισμού δεν γίνεται αντιληπτό. Μετά έχουμε Run Length Encoding και κωδικοποίηση της εντροπίας μέσω Huffman.

Παρακάτω γίνεται ανάλυση της υλοποίησης καθενός από τα παραπάνω βήματα. Παράλληλα παρουσιάζονται κάποια διαγράμματα που ζητούνται στην εκφώνηση της εργασίας. Στο τέλος της αναφοράς έχουμε ορισμένα συμπεράσματα για την ποιότητα του αποκωδικοποιημένου ήχου και τον βαθμό συμπίεσης.

II. Implementation

A. Subbands

Το πρώτο μέρος της εργασία επικεντρώνεται στη διαμέριση του φάσματος των δειγμάτων εισόδου. Η διαμέριση γίνεται χρησιμοποιώντας $M = 32$ θεωρητικά μη επικαλυπτόμενα φίλτρα. Τα φίλτρα υπολογίζονται μέσω της πρότυπης κρουστικής απόκρισης $h(n)$ 1 ως

$$h_i(n) = h(h)\cos\left(\frac{(2i+1)\pi}{2M}n + \frac{(2i+1)}{4}\pi\right) \qquad (1)$$

Μετασχηματίζοντας τις $h_i(n)$ στο πεδίο της συχνότητας χρησιμοποιώντας το διακριτό μετασχηματισμό Fourier μπορούμε σε δούμε τις αποκρίσεις των φίλτρων σε $dB(20\log(H(f)))$ 2. Χρήσιμη είναι και η απεικόνιση σε barks αντί για Hz γιατί δείχνει πιο αντιπροσωπευτικά τις αποστάσεις των διαφόρων συχνοτήτων όπως τις αντιλαμβάνεται η ανθρώπινη ακοή.

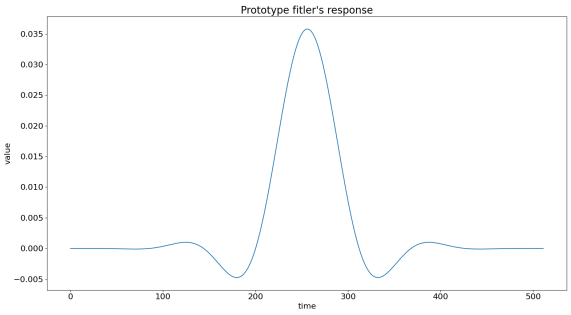

Figure 1. Prototype filter response

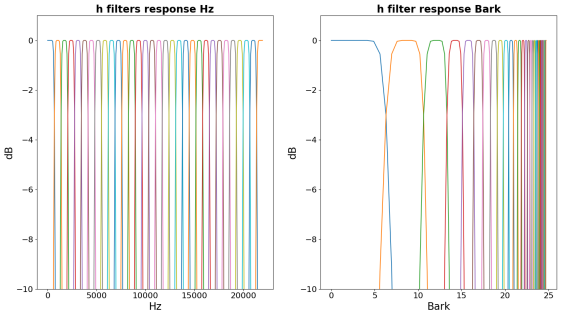

Figure 2. Filters' responses

Από τα παραπάνω, συμπερένουμε ότι τα φίλτρα είναι ζωνοπερατά με εύρος ζώνης $\frac{f_s}{2M} \approx 689Hz$

Το φιλτράρισμα των δειγμάτων ήχου γίνεται τμηματικά και παράγονται frames μεγέθους $N \times M$ όπου N τα δείγματα ενός subband και M το πλήθος των subbands. Κάθε τμήμα ήχου χρειάζεται να συνελιχτεί στο πεδίο του χρόνου με τις αποκρίσεις των φίλτρων. Η συνάρτηση ανάλυσης των subbands, μας παρέχετε έτοιμη και χρειάζεται μόνο να υλοποιήσουμε έναν buffer εισόδου. Το μέγεθός του είναι $(N-1)M + L$ και τα επιπλέον δείγματα χρειάζονται για να πραγματοποιηθεί ορθά η διαδικασία της συνέλιξης. Σε κάθε επανάληψη ο buffer προχωράει κατά NM δείγματα και

στο τέλος αν δεν επαρκεί ο αριθμός των δειγμάτων για τη συμπλήρωση ενός frame, προστίθενται μηδενικά.

Η ανάποδη διαδικασία είναι από τα frames να παράγουμε τα δείγματα εισόδου. Για την ανακατασκευή από τα subbands χρησιμοποιούνται τα φίλτρα G τα οποία είναι έτσι σχεδιασμένα, ώστε να επιτρέπουν πλήρη ανακατασκευή των δειγμάτων που φιλτραρίστηκαν με τα φίλτρα H. Όπως και η συνάρτηση ανάλυσης έτσι και η συνάρτηση σύνθεσης μας παρέχεται έτοιμη. Αυτή τη φορά το buffer εισόδου έχει μέγεθος $((N-1)+\frac{L}{2})\times M$ όπου L είναι το μέγεθος των φίλτρων σύνθεσης G. Επαναληπτικά διαβάζω τα frames και τα προσθέτω στην αρχή του buffer αφού έχω μετατοπίσει τα προηγούμενα στοιχεία του.

Όπως αναφέρθηκε, θεωρητικά οι δύο παραπάνω διαδικασίες, της ανάλυσης και της σύνθεσης θα πρέπει να επιτρέπου πλήρη ανακατασκευή. Παρόλα αυτά λόγω της υλοποίησης της συνέλιξης με buffers, χάνω κάποια από τα δείγματα. Παρόλα αυτά με κατάλληλη μετατόπιση η αρχική και η τελική χρονοσειρά είναι αρκετά κοντά όπως φαίνεται παρακάτω στα 3, 4. Επιπρόσθετα, ακουστικά δεν υπάρχει καμία αισθητή διαφορά μεταξύ του αρχικού και του ανακατασκευασμένου ήχου και το μέσο SNR του τελικού σήματος είναι $\sim 70dB$ αν δεν λάβουμε προφανώς τις τιμές που έχω μηδενικό σφάλμα και απειρίζεται το SNR.

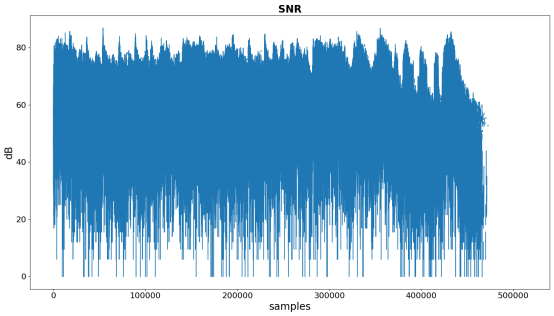

Figure 3. Reconstruction error SNR (dB) after inverting subband

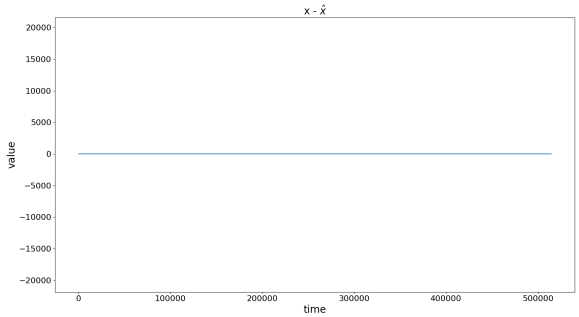

Figure 4. Reconstruction error $x - \hat{x}$ after inverting subband

Σε αυτή την υποενότητα καλούμασταν να υλοποιήσουμε μια αρχική εκδοχή του codec που θα κωδικοποιεί τα δεδομένα και μετά θα τα αποκωδικοποιεί. Δεδομένου ότι μέχρι στιγμής υλοποιήσαμε μόνο το subband filtering, η επιπρόσθετη επεξεργασία στην κωδικοποίηση και αποκωδικοποίηση του frame, γινόταν μέσω δύο κενών συναρτήσεων donothing και idonothing αντίστοιχα. Έπειτα μας ζητούσε η εκφώνηση να υλοποιήσουμε και δύο επιπλέον συναρτήσεις, το coder0 και decoder0 που είναι πρακτικά η κωδικοποίηση και αποκωδικοποίηση που υλοποιήσαμε στο codec0, απλά χωρισμένη σε δύο συναρτήσεις.

B. Discrete Cosine Transform

Αφού χωρίσουμε τα δείγματα ήχου σε frame και 32 subbands, μετά εφαρμόζουμε Discrete Cosine Transform (DCT) για μεταφορά από το χρόνο στη συχνότητα και περετέρω διαμέριση του φάσματος. Για να πάρουμε τους συντελεστές DCT αξιοποιήθηκε η συνάρτηση dct του scipy [2]. Εφαρμόζεται dct σε κάθε subband ενός frame ξεχωριστά και τοποθετούνται οι συντελεστές σειριακά, ο ένας μετά τον άλλον. Για την αντίστροφη διαδικασία χρησιμοποιήθηκε η

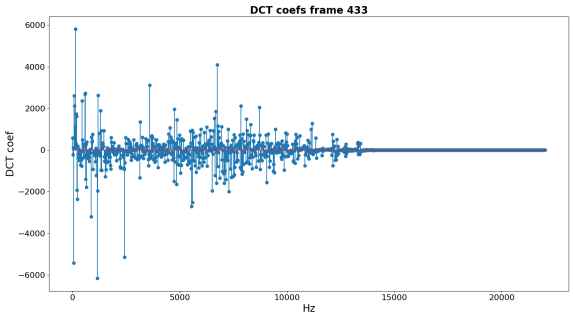

Figure 5. DCT coefficients of subbands example 1

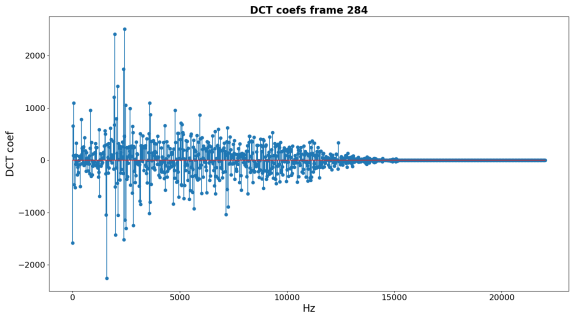

Figure 6. DCT coefficients of subbands example 2

συνάρτηση idct του scipy που μας μεταβαίνει στο πεδίο της συχνότητας. Ο αντίστροφος DCT γίνεται για τους συντελεστές κάθε subband και δημιουργείται εν τέλη το αρχικό frame χωρισμένο σε subbands στο πεδίο του χρόνου.

Για την επιβεβαίωση της ανακατασκευής από τη διαδικασία του DCT υπολογίζω το άθροισμα διαφοράς τετραγώνων του αρχικού και του τελικού frame. Παρατηρώ ότι έχω τέλεια ανακατασκευή.

C. Psychoacoustic Model

Πριν προχωρήσουμε στην επιλογή κβαντιστών για κάθε frame, χρειάζεται να υλοποιήσουμε το ψυχοακουστικό μοντέλο. Αυτό αξιοποιεί καταρχάς το κατώφλι ακουστότητας στη σιωπή που θέτει ένα όριο έντασης κάτω του οποίου οι συχνότητες δε γίνονται αντιληπτές από το ανθρώπινο αυτί. Το κατώφλι αυτό προέκυψε από πειράματα και δίνεται έτοιμο τόσο στο MP3 πρότυπο όσο και στην παρούσα εργασία. Πρακτικά στο μοντέλο, οι συχνότητες που δε γίνονται

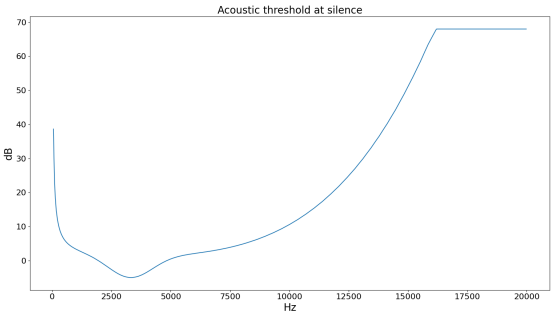

Figure 7. Acoustic threshold at silence

αντιληπτές από το ανθρώπινο αυτί της αποκωδικοποιεί με χαμηλότερης ποιότητας κβαντιστή, καθώς το σφάλμα που θα εισάγεται δε θα είναι έντονα αντιληπτό από τον άνθρωπο. Το μοντέλο αυτό εκμεταλλεύεται και την ιδιότητα ότι οι συχνότητες κοντά σε ισχυρούς τόνους επισκιάζονται από τον ισχυρό τόνο και δε γίνονται αντιληπτές. Επομένως, η ύπαρξη ισχυρών τόνων θα λέγαμε ότι αυξάνει το κατώφλι ακουστότητας σε μια γειτονική περιοχή συχνοτήτων.

Εξετάζοντας με τη σειρά τα ερωτήματα σε αυτή την ενότητα έχουμε αρχικά την τετριμμένο υπολογισμό της ισχύς των dct συντελεστών σε dB. Αυτό γίνεται ως

$$P(k) = 10\log(|c(k)|^2) = 20\log(|c(k)|) \tag{2}$$

c(k) είναι ο k συντελεστής dct ενός frame. Στη συνέχεια έχουμε την υλοποίηση του Dksparse που υπολογίζει κατά κάποιο τρόπο ένα πίνακα γειτνίασης. Συγκεκριμένα αν συμβολίσουμε με D το πίνακα αυτόν τότε το $D(k,j) = 1$ αν η συχνότητα j ανήκει στη γειτονιά του k. Το εύρος της γειτονιάς εξαρτάται από τη συχνότητα k.

$$\Delta_k \in \begin{cases} 2 & 2 < k < 282 \\ 2-13 & 282 \leq k < 570 \\ 2-13 & 570 \leq k < 1152 \end{cases} \tag{3}$$

Στη συνέχεια υλοποιήθηκε η συνάρτηση που αρχικοποιεί του ισχυρούς τόνους. Ισχυρός τόνος θεωρείται κάθε συντελεστής dct που έχει ισχύ μεγαλύτερη από τους άμεσους γειτόνους του και μεγαλύτερη κατά 7dB τουλάχιστον από τους γειτόνους που υπολογίστηκαν παραπάνω Δ_k. Για κάθε τόνο που εντοπίζεται εδώ παίρνω την ισχύ του μαζί με το άθροισμα των αμέσων γειτόνων του, προκειμένου να του αποδώσω τη συνολική ισχύ του που λόγω μειωμένης ευκρίνειας στο διακριτό φάσμα μπορεί να μοιράστηκε η ισχύς του στη γειτονιά του.

Για την επόμενη βασική λειτουργία του ψυχοακουστικού μοντέλου, χρειάζεται να υλοποιήσουμε πρώτα τη συνάρτηση που μετατρέπει μια σειρά συχνοτήτων σε Barks. Ο τύπος είναι

$$z(f) = 13\arctan(0.00076f) + 3.5\arctan((\frac{f}{7500})^2)(Bark) \tag{4}$$

Η επόμενη συνάρτηση που υλοποιούμε είναι η ST_reduction που έχει σκοπό να φιλτράρει τους masking tones που εντοπίστηκαν. Συγκεκριμένα κρατάει μόνο αυτούς πάνω από το κατώφλι ακουστότητας στη σιωπή 7. Επίσης από αυτούς που απέχουν λιγότερο από 0.5 barks κρατάει αυτόν με τη μεγαλύτερη ισχύ.

Ο κάθε τόνος ανάλογα με τη θέση του έχει διαφορετική επιρροή στο τελικό κατώφλι ακουστότητας. Πριν υπολογίσουμε την επιρροή του κάθε τόνου πρέπει να βρούμε πως απλώνεται στις διάφορες συχνότητες. Ο τύπος διασποράς δίνεται

$$SF(i,k) = \begin{cases} 17\Delta_z - 0.4P_M(k) + 11 & -3 \leq \Delta_z < -1 \\ (0.4P_M(k) + 6)\Delta_z & -1 \leq \Delta_z < 0 \\ -17\Delta_z & 0 \leq \Delta_z < 1 \\ (0.15P_M(k) - 17)\Delta_z - 0.15P_M(k) & 1 \leq \Delta_z < 8 \end{cases} \tag{5}$$

όπου k είναι η συχνότητα που βρίσκεται ο masking tone, i είναι η συχνότητα στην οποία μελετάμε την επιρροή του k τόνου και Δ_z είναι η διαφορά του i και του k τόνου σε barks.

Η εξάπλωση του κάθε τόνου στις διάφορες συχνότητες αξιοποιείται στον υπολογισμό της συνολικής επιρροής του κάθε masking tone στο κατώφλι ακουστότητας στη σιωπή. Ο τύπος που υπολογίζει την επιρροή είναι

$$T_M(i,k) = P_M(k) - 0.275z(f_k) + SF(i,k) - 6.025 \tag{6}$$

όπου k είναι ο τόνος του οποίου μελετάμε την επιρροή στη i συχνότητα. Έχοντας υπολογίσει την επίδραση του κάθε masking tone μπορούμε εύκολα να υπολογίσουμε στη Global_Masking_Thresholds το συνολικό acoustic threshold ως άθροισμα των ισχύων του κατωφλιού στην σιωπή και συνολικού κατωφλιού των τόνων σε κάθε συχνότητα. Εδώ κύρια ήθελε προσοχή ότι το άθροισμα δεν γινόταν σε dB αλλά το τελικό αποτέλεσμα μετατρέπονταν σε dB.

Συνοψίζοντας όσο αναφορά το ψυχοακουστικό μοντέλο, υλοποιούμε τον αλγόριθμο 1 που χρησιμοποιεί τις παραπάνω συναρτήσεις για να υπολογίσει το τελικό κατώφλι ακουστότητας ενός frame.

Δύο παραδείγματα από το κατώφλι ακουστότητας ενός frame είναι οι εικόνες 8 και 9. Στην πρώτη έχουμε μεγάλη

Algorithm 1 Psychoacoustic Model

Input: c, D, Tq

Output: Tg

1: $ST = STinit(c, D)$
2: $STred, STpower = STreduction(ST, c, Tq)$
3: $tm = MaskingThresholds(STred, STpower, Kmax)$
4: $Tg = GlobalMaskingThresholds(tm, Tq)$

επιρροή των masking tones ενώ στη δεύτερη καθορίζεται κυρίως από το κατώφλι ακουστότητας στη σιωπή.

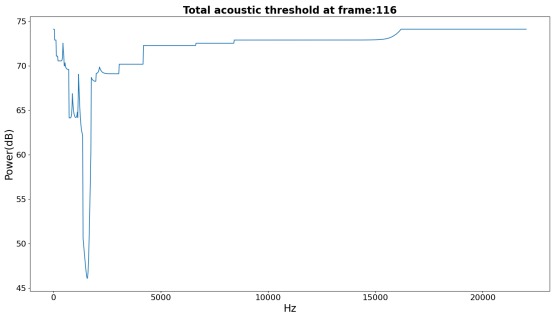

Figure 8. Total acoustic threshold example 1

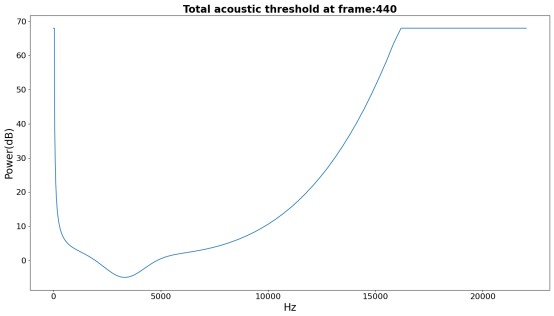

Figure 9. Total acoustic threshold example 2

D. Quantization

Μετά το ψυχοακουστικό μοντέλο, σειρά έχει η υλοποίηση του κβαντιστή. Αρχικά πρέπει να υπολογίσουμε στην critical_bands σε ποια κρίσιμη μπάντα ανήκει η κάθε συνάρτηση. Οι διάφορες ομάδες συχνοτήτων φαίνονται στο 10.

Το σύνολο των συχνοτήτων που βρίσκονται σε κάποια critical band κβαντίζονται με τον ίδιο κβαντιστή. Πριν κβαντιστούν όμως πραγματοποιείται μία κανονικοποίηση τους. Συγκεκριμένα, για τους συντελεστές κάθε μπάντας, υπολογίζεται οι συντελεστές κλίμακας της κάθε κρίσιμης ζώνης ως

$$Sc(band) = max(|c(i)|^{\frac{3}{4}}) \tag{7}$$

Figure 10. Critical Bands

Band No.	Center Frequency (Hz)	Bandwidth (Hz)
1	50	-100
2	150	100-200
3	250	200-300
4	350	300-400
5	450	400-510
6	570	510-630
7	700	630-770
8	840	770-920
9	1000	920-1080
10	1175	1080-1270
11	1370	1270-1480
12	1600	1480-1720
13	1850	1720-2000
14	2150	2000-2320
15	2500	2320-2700
16	2900	2700-3150
17	3400	3150-3700
18	4000	3700-4400
19	4800	4400-5300
20	5800	5300-6400
21	7000	6400-7700
22	8500	7700-9500
23	10500	9500-12000
24	13500	12000-15500
25	19500	15500-

και οι κανονικοποιημένοι συντελεστές υπολογίζεται ως

$$\tilde{c}(i) = sign(c(i)) \frac{|c(i)|^{\frac{3}{4}}}{Sc(band)} \tag{8}$$

οι κανονικοποιημένοι συντελεστές κβαντίζονται, ενώ οι συντελεστές κλίμακας αποθηκεύονται και χρησιμοποιούνται μετά κατά τον αποκβαντισμό.

Στη συνέχεια υλοποιώ τον κβαντιστή, ο οποίο παίρνει τον αριθμό των bits που θα αξιοποιηθούν και τους κανονικοποιημένους συντελεστές dct. Οι στάθμες του κβαντιστή δίνονται

$$\begin{aligned} d = [&-1, -(2^{b-1}-1)w_b, -(2^{b-1}-2), ..., \\ &-wb, wb, ..., (2^{b-1}-2)w_b, (2^{b-1}-1)w_b, 1] \end{aligned} \tag{9}$$

όπου $w_b = \frac{1}{2^b-1}$. Ο κβαντιστής βλέπουμε ότι δεν είναι ομοιόμορφος, οι δύο στάθμες εκατέρωθεν του μηδέν έχουν συγχωνευτεί σε μία και συνολικά έχει $2^b - 1$ ζώνες. Ο αποκβαντιστής αντιστρέφει την παραπάνω διαδικασία και προβάλει κάθε σύμβολο στο κέντρο της στάθμης που αντιστοιχεί.

Αφού υλοποιήθηκαν ο κβαντιστής και ο αποκβαντιστής πρέπει να υλοποιηθεί και η διαδικασία κβαντισμού της κάθε κρίσιμης μπάντας. Ανάλογα με το συνολικό κατώφλι ακουστότητας κάθε μπάντα θα κωδικοποιηθεί με διαφορετικό κβαντιστή, δηλαδή με διαφορετικό αριθμό bits. Για τον σκοπό αυτό για κάθε κρίσιμη μπάντα ξεκινάμε τη διαδικασία του κβαντισμού με ένα λίγα bit. Έπειτα γίνεται η διαδικασία του αποκβαντισμού και εκτιμάται το σφάλμα. Αν αυτό είναι κάτω από το κατώφλι ακουστότητας για τη συγκεκριμένη μπάντα, τότε επιλέγεται αυτός ο κβαντιστής αλλιώς αυξάνεται ο αριθμός των bits. Πρακτικά δε θέλουμε το τελικό σφάλμα κβαντισμού/αποκβαντισμού να είναι σημαντικά αισθητό στο ανθρώπινο αυτί.

Η αντίστροφή διαδικασία είναι σχετικά απλή αφού έχουν αποθηκευτεί τα bits του κάθε κβαντιστή της κάθε κρίσιμης μπάντας και οι scale factors των dct coefficients.

E. Run Length Encoding

Αφού υπολογίσαμε ένα σύμβολο για κάθε dct coefficient, στη συνέχεια εφαρμόζουμε run length encoding. Ο αλγόριθμος αυτός αναζητάει διαδοχικές επαναλήψεις μηδενικών και τις κωδικοποιεί με το αρχικό σύμβολο μαζί με έναν αριθμό που σηματοδοτεί πόσες φορές επαναλαμβάνεται το σύμβολο 0. Η συνάρτηση λοιπόν επιστρέφει ένα πίνακα με $R \times 2$ στοιχεία, όπου R τα μήκη διαδρομής που εντόπισε. Στην πρώτη στήλη αποθηκεύεται το πρώτο σύμβολο, ενώ στη δεύτερη το πόσες φορές επαναλήφθηκε το σύμβολο 0.

Η αντίστροφή διαδικασία γίνεται διαβάζοντας κάθε γραμμή του πίνακα που εξάγει η RLE και επαναλαμβάνοντας το στοιχείο 0 μετά το στοιχείο της πρώτης στήλης όσες φορές αναγράφει η δεύτερη στήλη.

Το συγκεκριμένο βήμα είναι χρήσιμο καθώς σε αρκετά frame έχουμε αρκετά επαναλαμβανόμενα μηδενικά. Παρόλο που αποθηκεύουμε και το σύμβολο και έναν επιπλέον αριθμό, προκύπτει ότι τελικά ο συνολικός αριθμός στοιχείων που χρειάζεται αν αποθηκεύσω είναι μικρότερος. Αυτό όμως προϋποθέτει όπως είπα να υπάρχουν αρκετά και μεγάλα μήκη διαδρομής, διαφορετικά αυτό το βήμα οδηγεί σε αύξηση των απαιτήσεων μνήμης.

F. Huffman

Εδώ επικεντρωνόμαστε στην κωδικοποίηση της εντροπίας. Ο κώδικας huffman απαιτεί τη δημιουργεία ενός binary tree. Γιαυτό υλοποιήθηκαν κάποιες υπορουτίνες που ορίζουν ένα node ενός δέντρου και το κατασκευάζουν ακολουθιακά ξεκινώντας από την κορυφή.

Αρχικά γίνεται εντοπισμός των διακριτών συμβόλων μαζί με τη συχνότητες εμφάνισή τους. Τα σύμβολα είναι δυαδικά καθώς υπάρχει το κβαντισμένο σύμβολο και ο αριθμός που επαναλαμβάνεται. Ο αλγόριθμος huffman κατασκευάζει το δέντρο συγχωνεύοντας επαναληπτικά τα nodes με τις μικρότερες συχνότητες εμφάνισης σαν παιδιά ενός νέου node με συχνότητα το άθροισμα των παιδιών node.

Αφού κατασκευαστεί το δέντρο αναζητάμε όλα τα μοναδικά σύμβολα και κάθε φορά που επισκεπτόμαστε αριστερό node προσθέτουμε 0 στη σειρά κωδικοποίησης του συμβόλου. Αντίθετα, αν επισκεπτόμαστε δεξιό κόμβο προσθέτουμε 1. Αυτή η διαδικασία οδηγεί στη δημιουργία ενός λεξικού με το οποίο αντιστοιχίζουμε κάθε σύμβολο με μια συμβολοσειρά 0 και 1. Τα σύμβολα που εμφανίζονται πιο συχνά έχουν μικρότερο αριθμό bit γιατί βρίσκονται υψηλότερα στο δυαδικό δέντρο αναζήτησης.

Η αντίστροφη διαδικασία γίνεται με την αξιοποίηση των συμβόλων και των αντίστοιχων συχνοτήτων που αποθηκεύσαμε στη διαδικασία του huffman. Με αυτό ανακατασκευάζουμε το δέντρο και την αντιστοιχία των συμβόλων σε bitstream 0 και 1 και διαβάζοντας ένα ένα τα bits του κωδικοποιημένου frame αναγνωρίζουμε σύμβολα. Αυτό μπορεί να γίνει γιατί δεν υπάρχει κωδικοποιημένο σύμβολο που να αποτελεί και πρόθεμα ενός άλλου κωδικοποιημένου συμβόλου.

G. MP3 codec

Σαν τελευταίο στάδιο έχουμε τον συνδυασμό των παραπάνω υπορουτίνων για την υλοποίηση μιας συνάρτηση κωδικοποιητή (coder) για την κωδικοποίηση σήματος σε MP3, αποκωδικοποιητή (decoder) για την αντιστροφή της προηγούμενες διαδικασίας και τέλος ενός codec για τον συνδυασμό coder και decoder.

Algorithm 2 MP3 coder simplified

Input: x
Output: bitstream

$bitstream = []$
$yframes = subband\_analysis(x)$
for $frame \in yframes$ **do**
  $dct\_c = dct(frame)$
  $Tg = psychoacoustic\_model(dct\_coef)$
  $dct\_quant = quantize(dct\_c, Tg)$
  $rle\_sym = RLE(dct\_quant)$
  $huff\_syms = huffman(rle_sym)$
  $bitstream.append(huff\_syms)$

Algorithm 3 MP3 decoder simplified

Input: bitstream
Output: x_hat

$yframes = []$
for $frame\_bits \in bitstream$ **do**
  $rle\_syms = inv\_huffman(frame\_bits)$
  $dct\_quant = inv\_RLE(rle\_syms)$
  $dct\_quant = dequantize(dct\_quant)$
  $yframe = inv\_dct(frame)$
  $yframes.append(yframe)$
$x\_hat = subband\_synthesis(yframes)$

Algorithm 4 MP3 codec simplified

Input: x
Output: x_hat

$bitstream = MP3coder(x)$
$x\_hat = MP3decoder(bitstream)$

Στα 2, 3, 4 παρατηρούμε μια απλουστευμένη εκδοχή (δεν περιλαμβάνονται όλες οι είσοδοι και έξοδοι) σε ψευδογλώσα που παρουσιάζει την κύρια ροή του encoder, decoder και codec.

Αξίζει να σημειωθεί ότι κατά τη διαδικασία εξαγωγής των huffman bits καθώς η συμβολοσειρά των 01 ήταν μεγάλη, όπως υποδεικνύονταν από την εκφώνηση, αποθηκευόταν σε ένα binary file. Επιπρόσθετα αποθηκευόταν σε ξεχωριστό αρχείο οι επιπλέον πληροφορίες που απαιτούνταν για την ανασύνθεση του ήχου από bitstream. Αυτά ήταν τα scale factors των dct coefficients, ο αριθμός bits που χρησιμοποιούταν για τον κβαντιστή κάθε κρίσιμης μπάντας, σε κάθε frame, ο

συχνοτικός πίνακας για την αναστροφή του huffman coding και τέλος αποθηκευόταν ο αριθμός των bits σε κάθε frame. Το τελευταίο δεν είναι γενικά μέρος του προτύπου αλλά το χρησιμοποίησα για να μπορώ με ευκολία να γνωρίζω πόσα bits να διαβάσω από το binary file, για κάθε frame.

Για την ορθότητα των παραπάνω ρουτίνων δημιουργήθηκε ένα demostration script με τρεις επιλογές. Η πρώτη είναι η codec επιλογή που κωδικοποιεί και αποκωδικοποιεί ένα ηχητικό σήμα και το αποθηκεύει, ενώ τυπώνει και το ποσοστό συμπίεσης. Για το ποσοστό συμπίεσης, πέρα από το bistream λαμβάνω υπόψη και τις επιπλέον πληροφορίες που αποθηκεύονται, θεωρώντας 16bits ανά αριθμό.

Η δεύτερη λειτουργία είναι ο coder που κωδικοποιεί μόνο το σήμα και αποθηκεύει το bitstream και την έξτρα πληροφορία σε ένα φάκελο. Το bitstream αποθηκεύεται σε binary επομένως το μέγεθος του αρχείου είναι ρεαλιστικό. Για την έξτρα πληροφορία την αποθηκεύω σαν dictionary με numpy επομένως καταναλώνει αρκετά περισσότερο χώρο και δεν είναι αντικειμενικό το μέγεθος του.

Τελευταία λειτουργία, που μπορεί να γίνει μόνο αφού έχει εκτελεστεί ο coder ή ο codec, είναι ο decoder που αποκωδικοποιεί και αποθηκεύει το σήμα σε PCM με $16bits/sample$. Για να το πραγματοποιείσει αυτό χρησιμοποιεί το αποθηκευμένο bitstream και την έξτρα πληροφορία.

III. Results

Για τα τελικά αποτελέσματα αξίζει να αναφέρουμε τις συμβάσεις που πάρθηκαν. Για τον υπολογισμό του μεγέθους του αρχικού αρχείο, ιοθετήθηκε τα $16bits/sample$ του PCM. Για το τελικό αρχείο πέρα από το bitstream, την έξτρα πληροφορία την θεωρούμε $16bits/number$. Βέβαια στο παρακάτω πίνακα φαίνεται και ο βαθμός συμπίεσης και χωρίς την έξτρα πληροφορία. Σαν βαθμός συμπίεσης θεωρούμε τον τύπο

$$CompressRatio = \frac{Uncompressed \quad Size}{Compressed \quad Size} \tag{10}$$

ενώ το ποσοστό χώρου που απελευθερώνεται υπολογίζεται ως

$$SpaceSaving = 1 - \frac{Compressed \quad Size}{Uncompressed \quad Size} \tag{11}$$

Ακουστικά η αλείωση δεν είναι σημαντικά αισθητή. Φυσικά για να γίνει αυτό ήταν απαραίτητη η μετατόπιση του Tg προς τα κάτω κατά 12-15dB. Μια σχέση του compress ration με την μετατόπιση του Tg φαίνεται στο III. Ενδεικτικά το τελικό μέσο SNR του σήματος είναι $\sim 10dB$ για μετατόπιση του Tg ίση με -13dB.

Όπως αναγραφόταν στην εκφώνηση αυτό ήταν αναμενόμενο καθώς υλοποιήσαμε μια απλουστευμένη εκδοχή του MP3. Όσο πιο κάτω πάμε την καμπύλη τόσο περισσότερο αυστηρός είναι ο κβαντιστής και κωδικοποιεί με περισσότερα bits το σήμα. Το πλεονέκτημα είναι ότι ακούγεται καλύτερα το τελικό ανακτασκευασμένο σήμα, αλλά το ποσοστό συμπίεσης μειώνεται.

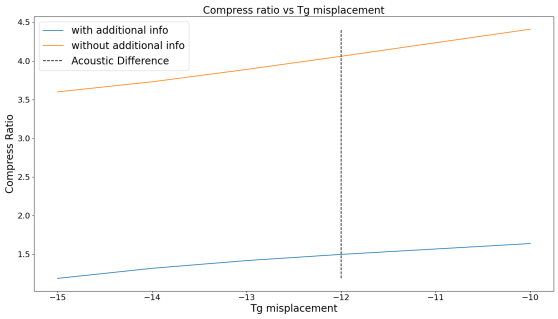

Figure 11. Compress ratio for different Tg displacement

Τέλος όσο αναφορά τον χρόνο εκτέλεσης, ένας μέσος χρόνο για την κωδικοποίση είναι 1 λεπτό και για την αποκωδικοποίηση 20 δευτερόλεπτα.

IV. Conclusion

Κλείνοντας θα λέγαμε ότι η υλοποίηση ενός MP3 πρωτοκόλλου έχει μια βασική δομή που στηρίζεται σε γενικές ιδέες της συμπίεσης δεδομένων όπως αποσυσχέτιση δεδομένων, συμπίεση και κωδικοποίηση εντροπίας. Διαφοροποιείται από άλλες συμπιέσεις από άλλες μορφές εισόδου όπως εικόνα, γιατί με την αξιοποίηση ψυχοακουστικού μοντέλου για τον έλεγχο του σφάλματος κβαντισμού. Η υλοποίηση αυτής της εργασίας αν και αποτελεί μια απλουστευμένη εκδοχή του MP3 πετυχαίνει αξιοπρεπή συμπίεση με μικρό ακουστικό κόστος στην ποιότητα ήχου.

References

[1] F. I. for Integrated Circuits (IIS), "Mpeg audio layer-3 (mp3) coding technology," Fraunhofer Institute for Integrated Circuits (IIS), Tech. Rep., 1991.

[2] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.