Technical University of Crete, ECE

Digital Image Processing
Project Report

Τσιαούσης Χρήστος
2016030017

Πρωτοπαπαδάκης Γιώργος
2016030134

January 5, 2021

1 Σκοπός

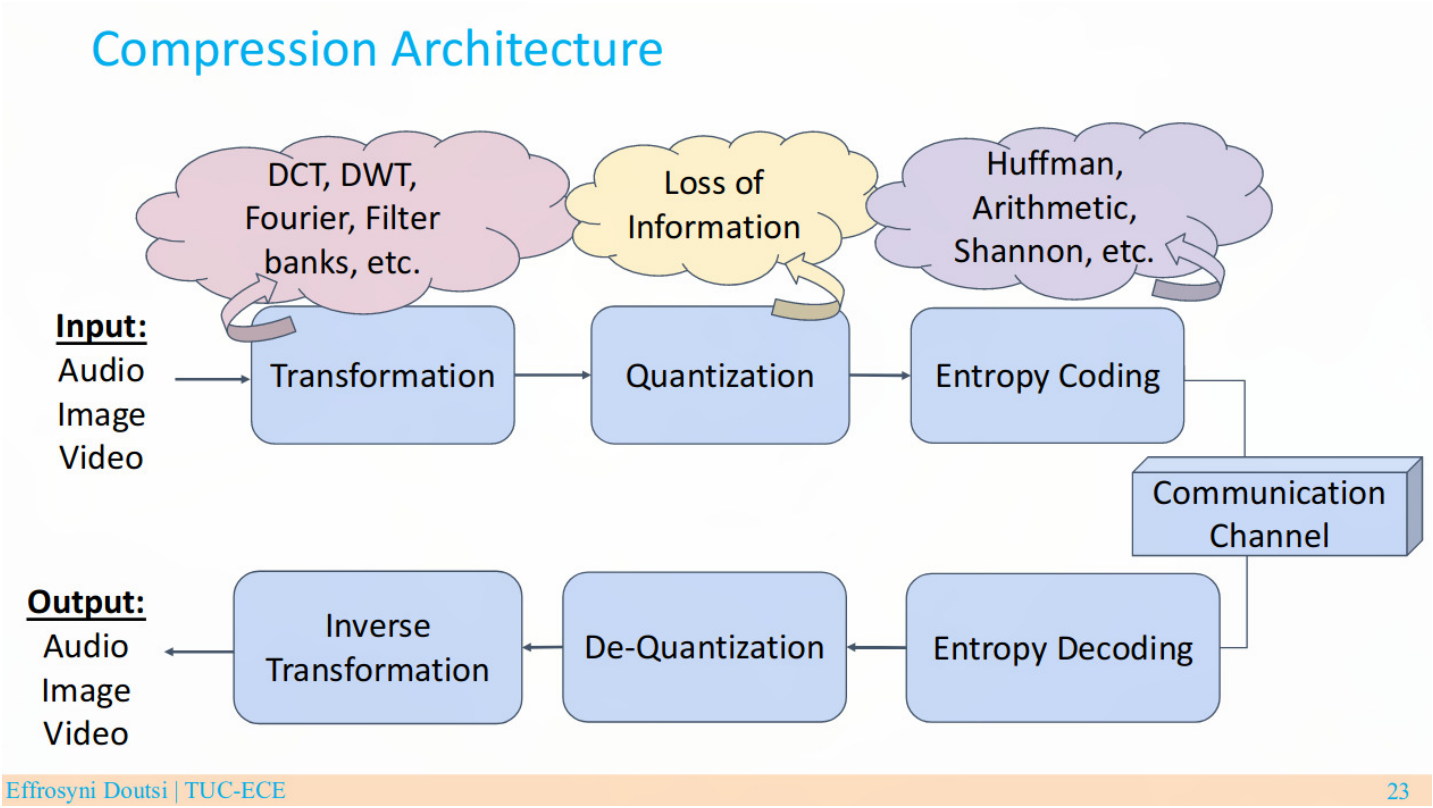


Figure 1.1: T/R pipeline.

**Ο Κώδικας βρίσκεται και συγκεντρωτικά στο τελευταίο κεφάλαιο*

Σκοπός του project ήταν η κατανόηση του παραπάνω pipeline, του όγκου πληροφορίας που κρύβεται στα multimedia και η υλοποίηση βασικών τεχνικών κβάντισης και κωδικοποίησης έτσι ώστε να ελαχιστοποιήσουμε κατά πολύ την μεταδιδόμενη πληροφορία.

2 Μέρος Πρώτο

2.1 Σε μονοδιάστατα σήματα

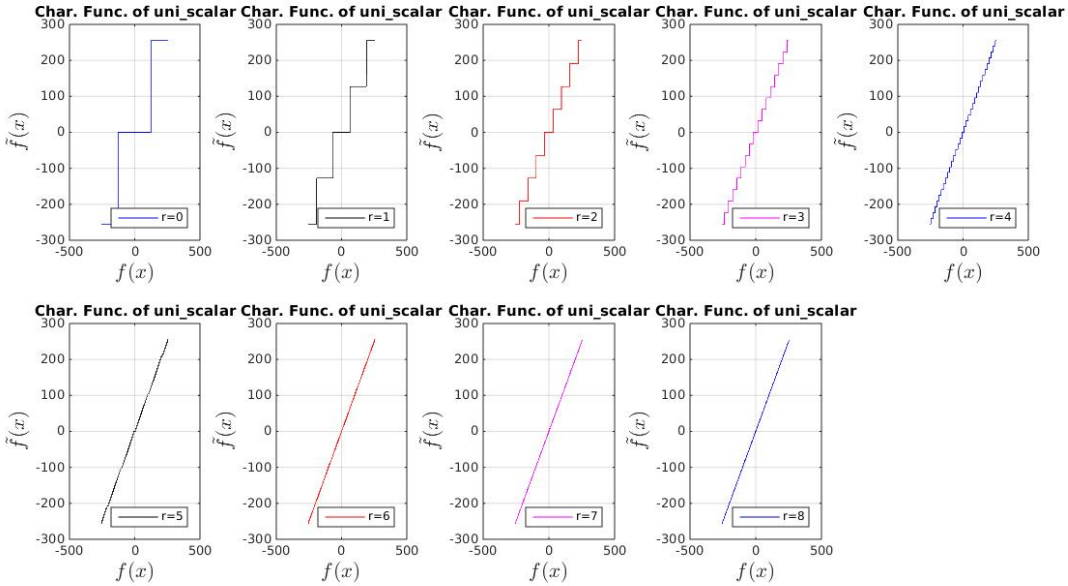


Figure 2.1: Η Χαρακτηριστική συνάρτηση του κβαντιστή.

Όπως φαίνεται και από την χαρακτηριστική συνάρτηση 2.1, υλοποιήσαμε έναν **mid-tread quantizer** και του δώσαμε ένα πολύ πυκνό (με βήμα 0.01) σήμα εισόδου, έτσι ώστε να φανεί η περιοχή deadzone. Η υλοποίηση ήταν απλή εφαρμογή του τύπου που δίνεται στην εκφώνηση και βρίσκεται εδώ.

2.2 Σε δισδιάστατα σήματα

Για την επιτυχή κβάντιση της εικόνας, υλοποιήσαμε την περίπτωση του δισδιάστατου σήματος στην **uni_scalar** και κάναμε την εξής παρατήρηση:
Αφού η εικόνα μας είναι grayscale και 8bit αντιστοιχούν για κάθε pixel, αυτό σημαίνει ότι το άνω όριο στις τιμές των pixel είναι 2^8-1. Αυτό σημαίνει ότι θα πρέπει να "μεταφραστεί" όπως φαίνεται παρακάτω.

$$A \in [-127, 127] \Rightarrow \boxed{A \in [0, 255]}$$

Γι αυτό τον λόγω λοιπόν, καλούμε την uni_scalar με όρισμα A/2 και όχι A. Τέλος μπορούμε να δούμε οπτικά τα επίπεδα κβάντισης να αυξάνονται, για παράδειγμα, η πρώτη εικόνα στο 2.2 έχει ένα απόλυτο επίπεδο, που σημαίνει ένα στα θετικά κι ένα στα αρνητικά. Αυτό έπειτα μεταφράζεται βάσει της παραπάνω σχέσης και δίνει τις τιμές που μας επιβεβαιώνει και το *colorbar:* 0 και 255.

2.3 Σφάλματα

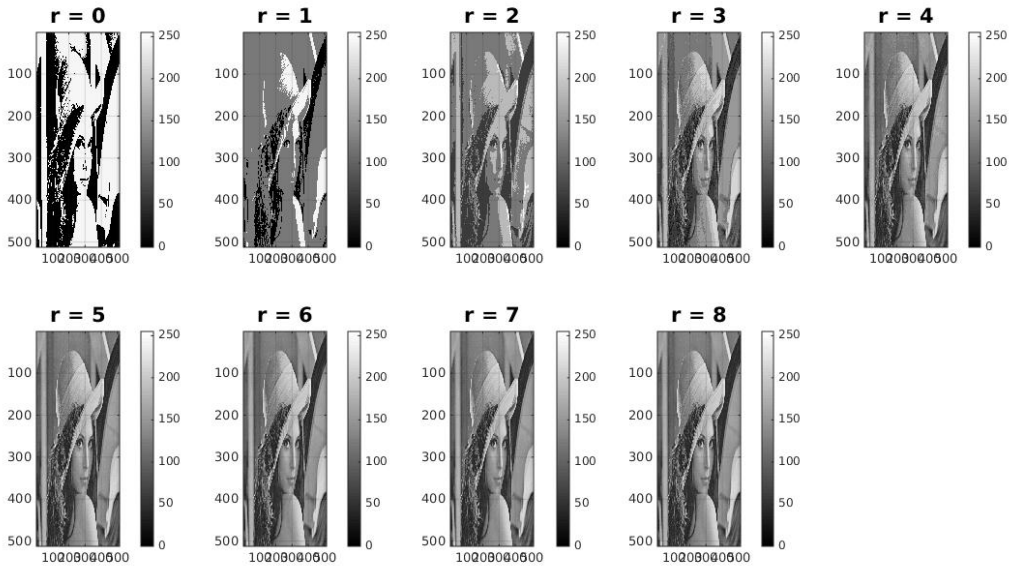


Figure 2.2: Η αγαπημένη Λένα κβαντισμένη σε κάθε configuration.

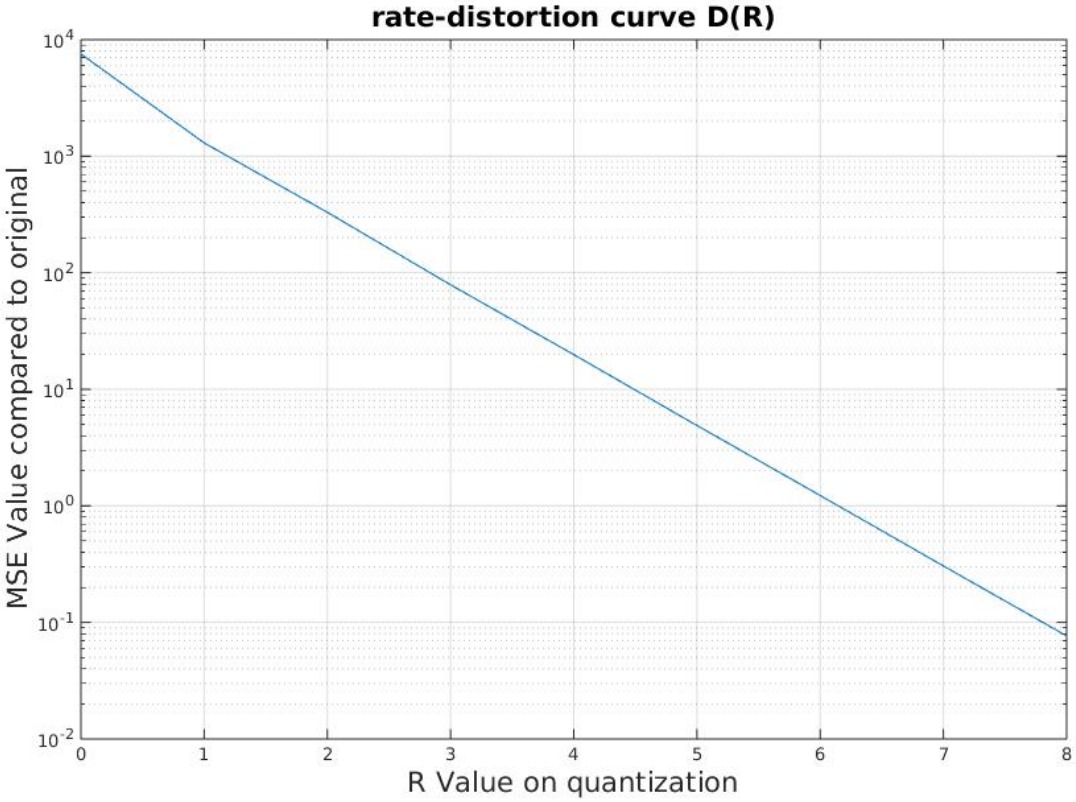


Figure 2.3: Το μέσο τετραγωνικό σφάλμα, ανά επίπεδο κβάντισης.

Το διάγραμμα των σφαλμάτων 2.3 είναι σε λογαριθμική κλίμακα. Με αυτό τον τρόπο βλέπουμε πολύ πιο εύκολα τις μεγάλες διακυμάνσεις στις τιμές. Ακόμα και για βήμα κβάντισης ίσο με την μονάδα

$$Q_{step} = 1 \Leftrightarrow (A = 127, L = 2^8),$$

όπου έχουμε προσεγγίσει κατά πολύ το 0, αλλά δεν το φτάνουμε ποτέ. Αυτό κατά πάσα πιθανότητα ωφείλεται στην **περιοχή deadzone**.
Η υλοποίηση του πρώτου μέρους βρίσκεται εδώ.

3 Μέρος Δεύτερο

Το βίντεο έχει **141 frames**, με ρυθμό **30 fps**, η ανάλυση του κάθε ενός είναι **320x240** και τέλος η διάρκεια **4.7 seconds**. Οι αριθμοί επιβεβαιώνονται μεταξύ τους και όποιο frame και να "πάρουμε", έχει πάντα την ίδια ανάλυση. Από άποψη υλοποίησης, δεν χρησιμοποιήθηκε η *imshow*, γιατι δεν έκανε σωστό allign στους άξονες του figure.

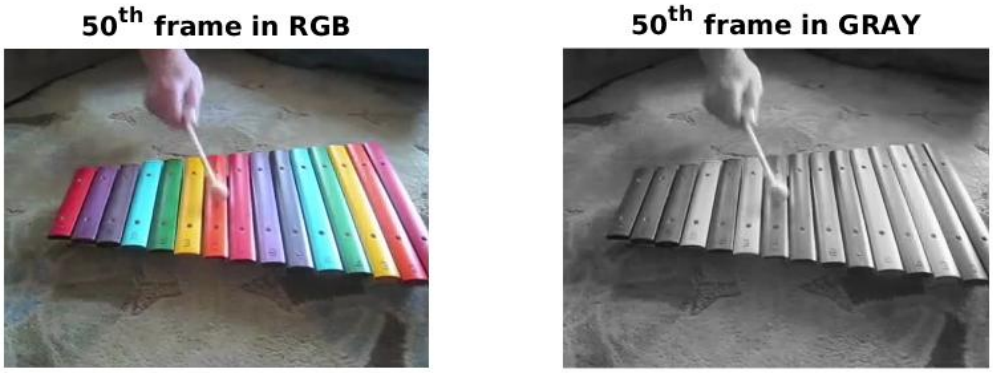


Figure 3.1: Frame number 50.

Αυτό που έχει μεγάλο ενδιαφέρον ως προς τον σχολιασμό, είναι ότι αφού κάναμε την αναπαραγωγή του video, έπρεπε να γίνει reinitialize το αντικείμενο για να πάμε σε συγκεκριμένο frame. Η πρώτη σκέψη είναι "Μα καλά δεν μπορούσαν να το υλοποιήσουν;", αλλά οι αστοχίες που υπήρχαν με την κλάση *VideoReader*, μας έδωσαν hints για όλο το backend που χρησιμοποιείται στην αναπαραγωγή βιντεο. Οι βιβλιοθήκες ffmpeg και GStreamer είναι ο ελβετικός σουγιάς των multimedia. Παρέχουν API για πολλές γλώσσες προγραμματισμού καθώς και εφαρμογές για terminal. Οι δυνατότητες είναι ατελείωτες. Με την απλή εντολή `ffmpeg -i xylophone3.mp4` μπορούμε να δούμε αναλυτικά τα metadata του αρχείου και πως έχει video encoding **h264**.

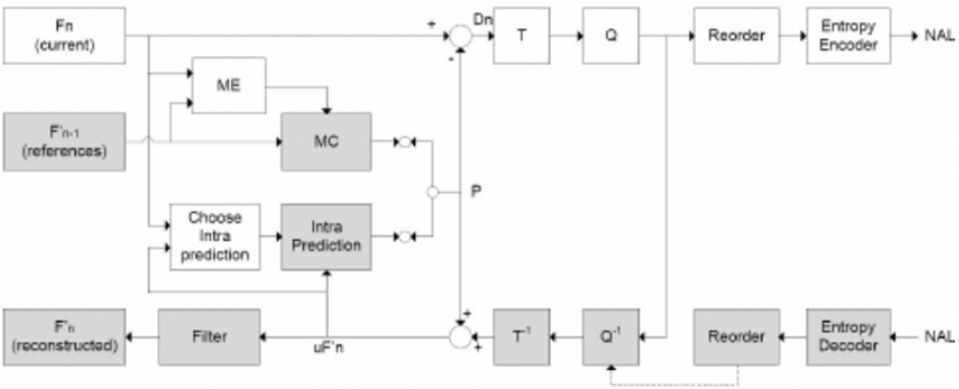


Figure 3.2: h264 encoder/decoder pipeline.

Πολύ συχνά στα βιντεο υπάρχει ένα **σταθερό background** και ένα πολύ μικρό ποσοστό των pixel αλλάζουν από frame σε frame. Αυτό ακριβώς εκμεταλεύεται αυτή η κωδικοποίηση και δεν μπορείς να μάθεις την πληροφορία ολόκληρου του frame χωρίς να ξέρεις το προηγούμενο. **Γι αυτό πρέπει να γίνει reinitialize το αντικείμενο!**. Τέλος στο 3.2 φαίνεται το pipeline της κωδικοποίησης αυτής, που προς θετική μας έκπληξη, απέχει ελάχιστα από το “βασικό” που κάνουμε σε αυτό το project.
Για να εκτιμηθεί η αξία του, ενδεικτικά το βίντεο που είχαμε να επεξεργαστούμε σε raw μορφή θα έπιανε περίπου 87 εκατομμύρια bit, ενώ τώρα είναι 1,2 εκατομμύρια bit!!
Δείτε τον κώδικα αυτού του μέρους εδώ.

4 Μέρος Τρίτο

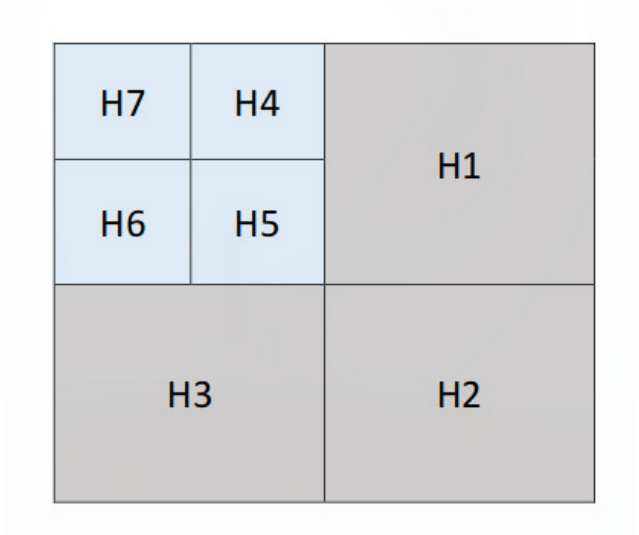
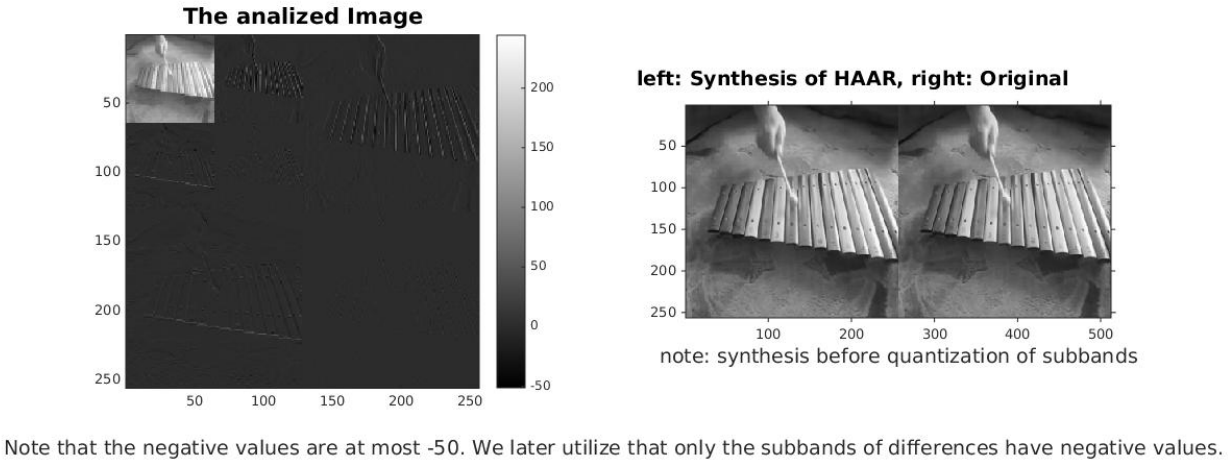


Figure 4.1: 2 level HAAR wavelet tansform for 2-D signal.

Σε αυτό το μέρος συνδυάσαμε τον κβαντιστή από το πρώτο μέρος και εξελήξαμε την μέθοδο Haar, που έχει υλοποιηθέι σε προηγούμενο εργαστήριο, ώστε να λειτουργεί με 2-D σήματα. Ο μετασχηματισμός αυτός χωρίζει την εικόνα σε τέσσερα τεταρτημόρια ανά επίπεδο. Κάθε επίπεδο παίρνει το άνω αριστερά του προηγουμένου για να κάνει την επεξεργασία, όπως έχουμε δει και στην θεωρία. Κάναμε έτσι μετασχηματισμό δυο επιπέδων στο grayscaled πεντηκοστό frame του ‘xylophone3.mp4’ όπως φαίνεται στο 4.2.



Note that the negative values are at most -50. We later utilize that only the subbands of differences have negative values.

Figure 4.2: 2 level HAAR wavelet tansform for 2-D signal.

(Υπενθύμιση: όλος ο κώδικας στο τελευταίο κεφάλαιο)

4.1 Αποφάσεις πριν την Κβάντιση

Όπως έχει επισημανθεί και επάνω στο 4.2, μπορεί κανείς να παρατηρήσει ότι οι τιμές στα τεταρτημόρια $H6 \rightarrow H1$ κυμαίνονται από -50 εώς και 255 (στα σημεία που είναι λευκά). Κι έτσι πρέπει να δώσουμε μεγαλύτερο παράθυρο κβάντισης από το A/2 που είδαμε στο πρώτο μέρος. Αν και το ένστικτο λέει ότι χοντρικά το A για την κβάντιση των subbands θα πρέπει να είναι

```
abs( min(min(subband)) + max(max(subband)) ) / 2,
```

πειραματικά είδαμε πως το καλύτερο σφάλμα σηματοθορυβικής σχέσης συγκριτικά με την αρχική εικόνα το πήραμε για A ίσο με την ελάχιστη τιμή pixel ολόκληρης της εικόνας και ίσως αν δεν δίναμε και μαθήματα σε πέντε μέρες να βρίσκαμε και βέλτιστη τιμή για το επόμενο βήμα (που έχουμε διαφορετικά επίπεδα κβάντισης). Τέλος να πούμε ότι ελέγχεται η σωστή σύνθεση haar με μέσο τετραγωνικό σφάλμα (που βγαίνει 0).

4.2 Εντροπία

Έπειτα υπολογίσαμε την εντροπία του κάθε subband ξεχωριστά, αλλά και ολόκληρης της εικόνας με την βοήθεια των συναρτήσεων που υλοποιήσαμε. Η συνολική εντροπία έπειτα από την κωδικοποίηση και την κβάντιση έπεσε στό **1.257**(!) κι έτσι βρήκαμε λόγο εντροπιών αρχικής και τελικής εικόνας **6.02**.

```
--------------PART C--------------

        --First Method--

ENTROPY--H7 :   7.475305e+00
ENTROPY--H6 :   7.481688e-01
ENTROPY--H5 :   4.092201e-01
ENTROPY--H4 :   1.028007e+00
ENTROPY--H3 :   4.458296e-01
ENTROPY--H2 :   1.941334e-01
ENTROPY--H1 :   6.658515e-01
ENTROPY--H0 :   1.257337e+00    (whole Image)
First Method: Entropy ratio is: 6.02
First Method: Compression ratio is: 1.88 (bit ratio)

        --Second Method--

ENTROPY--H7 :   7.478556e+00
ENTROPY--H6 :   1.186817e+00
ENTROPY--H5 :   7.201337e-01
ENTROPY--H4 :   1.482521e+00
ENTROPY--H3 :   1.873981e-01
ENTROPY--H2 :   6.345809e-02
ENTROPY--H1 :   3.999942e-01
ENTROPY--H0 :   1.242370e+00    (whole Image)
Second Method: Entropy ratio is: 6.09
Second Method: Compression ratio is: 2.17 (bit ratio)
---------------------------------
fx >>
```

Figure 4.3: Εντροπία από την κονσόλα του Matlab.

[Spoiler Alert] παραθέτουμε και του output της κονσόλας του matlab για τις τιμές της εντροπίας του κάθε subband στην 4.3.

4.3 Ψάχνωντας τα bits

Θέλαμε όμως να βρούμε και την πραγματική συμπίεση. Ψάχνοντας λίγο είδαμε ότι η συνάρτηση *whos()* δίνει πληροφορίες για οποιαδήποτε κλάση στο matlab, αλλά δεν μας έκανε επειδή δεν μπορέσαμε να κάνουμε μεταβλητά τα bit για κάθε pixel (no memory padding). Έτσι τα υπολογίσαμε

“χειροκίνητα” βρίσκοντας τα bit της αρχικής εικόνας με την σχέση $m \cdot n \cdot 8$ και τα bit της τελικής με την σχέση

$$(\frac{m}{4} \cdot \frac{n}{4} \cdot 8) + 3(\frac{m}{4} \cdot \frac{n}{4} \cdot QBits_{lvl2}) + 3(\frac{m}{2} \cdot \frac{n}{2} \cdot QBits_{lvl1}),$$

όπου ο πρώτος όρος αντιστοιχεί στο H7, ο δεύτερος στα H6, H5, H4, ο τρίτος στα H3, H2, H1 και τα QBits καμία σχέση με την κβαντομηχανική.
Έτσι, καταλήξαμε σε **πραγματικο ρυθμό συμπίεσης 1.88** για την πρώτη περίπτωση. Πράγμα το οποίο σημαίνει πως αν μας ενδιέφερε η αποστολή της εικόνας σε ένα τηλεπικοινωνιακό κανάλι θα είχαμε μεγαλύτερες ταχύτητες καθαρά και μόνο από τα bit κωδικοποίησης αλλά και μεγαλύτερη ανοχή στον θόρυβο λόγο κβάντισης. Η εικόνα που παρήγαγε η μεθοδολογία μας είναι η 4.4.

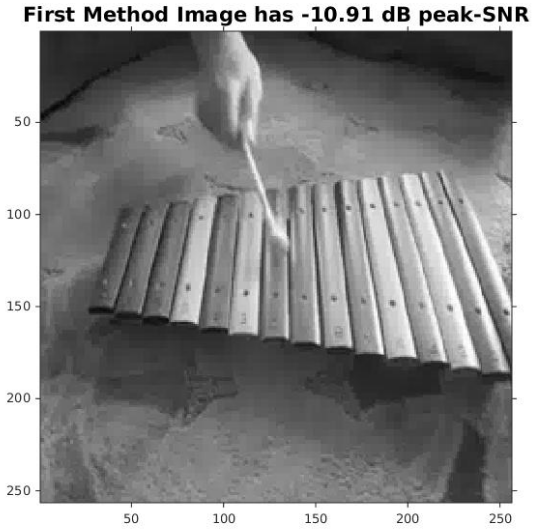


Figure 4.4: [part C] method 1.

Τελείως αντίστοιχα, η δεύτερη μέθοδος όπου κβαντίζει με 3bit το πρώτο επίπεδο και με 5 το δεύτερο έδωσε ελαφρώς διαφορετικό αποτελέσμα. Δηλαδή **6.09 συμπίεση εντροπίας και 2.17 συμπίεση bit**. Αυτό συνέβει επειδή διαλέξαμε λιγότερα bit αναπαράστασης της low-pass πληροφορίας και σταδιακά όλο και μεγαλύτερα όσο πηγαίναμε προς την εικόνα χαμηλής ανάλυσης (H7).
Τέλος βλέπουμε ότι η σηματοθορυβική σχέση είναι αρνητική. Αυτο ωφείλεται στο ότι για πολλά pixel οι τιμές διαφέρουν ελαφρώς και ο λόγος σήματος προς τον θόρυβο είναι μικρότερος της μονάδας, άρα και ο λογάριθμος αρνητικός. Παρ‘όλ’αυτά δεν τα πήγαμε και τόσο άσχημα, ειδικά αναλογιζόμενοι πως δεν κάναμε de-Quantization. Και πάνω σε αυτό να πούμε ότι αν και η δεύτερη μέθοδος έχει χειρότερο psnr metric, στο μάτι φαίνεται αρκετά καλύτερη και λιγότερο “pixeliασμένη”. Αυτό μας επιβεβαιώνει ότι οι μετρήσεις σφαλμάτων είναι μόνο ένας δείκτης κι όχι τι αντιλαμβάνεται το μάτι.
Κλείνοντας να πούμε ότι το project ήταν πολύ ενδιαφέρον, όπως και όλα τα υπόλοιπα εργαστήρια του μαθήματος, και σίγουρα μας έκαναν να βλέπουμε την επεξεργασία εικόνας με μια διαφορετική ματιά.

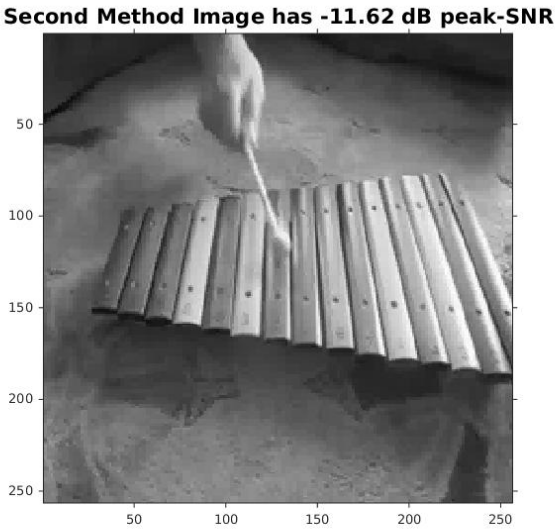


Figure 4.5: [part C] method 2.

5 Κώδικας

Οι υπογραμμισμένες συναρτήσεις είναι αυτές που έχουν υλοποιηθεί στα πλαίσια του project.

Listing 1: Η main.

```
clc;
clear all;
close all;


partA();
frame50 = partB('xylophone3.mp4');
partC(frame50);

return;
```

Listing 2: Η υλοποίηση του κβαντιστή.

```
% mid-tread quantizer
function D = uni_scalar(iSignal, inA, desiredL)
    assert(desiredL<=2^8,'input signal must be in [-255,255] range')

    step = (2*inA)/desiredL;

    [m, n] = size(iSignal);
    if( n == 1 ) %signal is 1-D
        D=zeros(m,n);
        for i=1:m
            roundVal = floor( (abs(iSignal(i))/step) + (1/2));
            D(i)= step * sign(iSignal(i)) * roundVal;
        end

    else %signal is 2-D
        D=zeros(m,n);
        for j=1:m
            for k=1:n
                roundVal = floor( (abs(iSignal(j,k))/step) + (1/2) );
                D(j,k)= step * sign(iSignal(j,k)) * roundVal;
            end
        end

    end
end
```

Listing 3: Η υλοποίηση της δισδιάστατης ανάλυσης Haar.

```
function OUT_IMAGE = haar2anal( imIn, level )
    [rows,columns] = size(imIn);
    assert((rows>1 && columns>1),'please input an image');
    OUT_IMAGE = double(imIn);
%% this section wiil apply for the upper left subband of previous level
    for lvl=1:level
        %lvl=1 applies to whole image
        % bigger levels apply to image/2^(level-1)

        %first rows
        for i=1:rows/(2^(lvl-1))
            tmp = OUT_IMAGE(i, 1:columns/(2^(lvl-1)));
            OUT_IMAGE(i, 1:columns/(2^(lvl-1))) = haar1analysis(tmp,1);
        end
        %then columns, notice the transpose signs (')
        for l=1:columns/(2^(lvl-1))
            tmp = OUT_IMAGE(1:rows/(2^(lvl-1)), l)';
            OUT_IMAGE(1:rows/(2^(lvl-1)), l) = haar1analysis(tmp,1)';
        end
    end
```

```
21
22 %% this commented section wiil apply for the whole image per level
23 %     for dummy=1:level
24 %         for i=1:rows
25 %             tmp = OUT_IMAGE(i,:);
26 %             OUT_IMAGE(i,:) = haar1analysis(tmp,1);
27 %         end
28 %         for l=1:columns
29 %             tmp = OUT_IMAGE(:,l)';
30 %             OUT_IMAGE(:,l) = haar1analysis(tmp,1)';
31 %         end
32 %     end
33 end
```

Listing 4: Η υλοποίηση της δισδιάστατης σύνθεσης Haar.

```
1  function OUT_IMAGE = haar2syn( imIn, level )
2      [rows,columns] = size(imIn);
3      assert((rows>1 && columns>1),'please input an image');
4      OUT_IMAGE = double(imIn);
5  %% this section wiil apply for the upper left subband per level
6      for lvl=level:-1:1
7          %lvl=1 applies to whole image
8          % bigger levels apply to image/2^(level-1)
9
10         %first rows
11         for i=1:rows/(2^(lvl-1))
12             tmp = OUT_IMAGE(i, 1:columns/(2^(lvl-1)));
13             OUT_IMAGE(i, 1:columns/(2^(lvl-1))) = haar1synthesis(tmp,1);
14         end
15         %then columns, notice the transpose signs (')
16         for l=1:columns/(2^(lvl-1))
17             tmp = OUT_IMAGE(1:rows/(2^(lvl-1)), l)';
18             OUT_IMAGE(1:rows/(2^(lvl-1)), l) = haar1synthesis(tmp,1)';
19         end
20     end
21
22 %% this commented section wiil apply for the whole image per level
23 %     for dummy=1:level
24 %         for i=1:rows
25 %             tmp = OUT_IMAGE(i,:);
26 %             OUT_IMAGE(i,:) = haar1synthesis(tmp,1);
27 %         end
28 %         for l=1:columns
29 %             tmp = OUT_IMAGE(:,l)';
30 %             OUT_IMAGE(:,l) = haar1synthesis(tmp,1)';
31 %         end
32 %     end
33 end
```

Listing 5: Η υλοποίηση της μονοδιάστατης ανάλυσης Haar.

```
1  function Ahaar = haar1analysis (A, level)
2      assert(level<=log2(length(A)),'Level too big for this input.');
3      assert((-1)^(length(A))==1,'Length of input signal must be even');
4
5      for decomp=1:level
6          assert((-1)^(length(A))==1,'Level length not even. Change value');
7          %% 1.b
8          k=1;
9          for i=1:2:(length(A)-1)
10             Ag{k} = A(:,i:i+1); %make tuples
11             k=k+1;
12         end
13         %% 1.c
14         for j=1:(length(A)/2) % length(Ag) doesn't work for multi-level
15            Am(j) = mean(Ag{j}); %calculate means
16         end
```

```
17
18          %% 1.d
19          p=1;
20          for x=1:2:(length(A)-1)
21              Ad(p) = A(x) - Am(p); %calculate diffs
22              p = p + 1;
23          end
24          %% 1.e
25          Ahaar(1:(length(A)/2)) = Am; %first half is means
26          Ahaar((length(A)/2)+1:length(A)) = Ad; %second is diferences
27          A = Am; % next level computed by mean values
28          clear Ad; %clear local variables
29          clear Am; %to have the right lengths
30          clear Ag; %for the next itteration
31      end
32  end
```

Listing 6: Η υλοποίηση της μονοδιάστατης σύνθεσης Haar.

```
1   function Arec = haar1synthesis (Ahaar , level)
2       assert(level<=log2(length(Ahaar)),'Level too big for this input.');
3       assert((-1)^(length(Ahaar))==1,'Length of input signal must be even');
4
5       Arec = Ahaar; % for multilevel calculations
6       for decomp=level:-1:1
7           Alevel=Arec(1:(end/2)/2^(decomp-1)); %means of level
8           %% 2.b
9           Au = Alevel; %upsampling the means of the specific level
10          Aupc = repmat(Au(:)',2,1); %repeating each value 2 times
11          Aupc = Aupc(:); %making the 2-d array, 1 column
12          Aup = Aupc.'; %making the column 1 line
13          %% 2.c
14          dif = Arec(length(Alevel)+1:end); % take all remaining diffs
15          q=1; % 2^(decomp-1);
16          for n=1:2:length(Aup)-1
17              Arec(n) = Aup(n) + dif(q);
18              Arec(n+1) = Aup(n+1) - dif(q);
19              q = q + 1;
20          end
21          clear dif; %clear local variables
22          clear Aup; %to have the right lengths
23          clear Au; %for the next itteration
24          clear Alevel;
25      end
26  end
```

Listing 7: Η υλοποίηση υπολογισμού εντροπίας κατά Shannon.

```
1   function en = imEntropy( I )
2       % 8-bits per pixel [0,255]
3       I = uint8(I);
4       % get image's histogram
5       [cnt,binsX] = imhist(I);
6       % remove zero entries that would cause NaN
7       cnt(cnt==0) = [];
8       % normalize histogram to propabilities
9       cnt = cnt/numel(I);
10      % calculate entropy
11      en = -sum(cnt.*log2(cnt));
12  end
```

Listing 8: Υπολογισμός εντροπίας για κάθε sub band ξεχωριστά.

```
1   function entrOfSubBands( imHarAnal )
2       [r,c] = size(imHarAnal);
3
```

```
    enH7 = imEntropy( imHarAnal(1:r/4  , 1:c/4) );
    enH6 = imEntropy( imHarAnal(r/4:r/2, 1:c/4) );
    enH5 = imEntropy( imHarAnal(r/4:r/2, c/4:c/2) );
    enH4 = imEntropy( imHarAnal(1:r/4  , c/4:c/2) );
    enH3 = imEntropy( imHarAnal(r/2:r  , 1:c/2) );
    enH2 = imEntropy( imHarAnal(r/2:r  , c/2:c) );
    enH1 = imEntropy( imHarAnal(1:r/2  , c/2:c) );
    enH0 = imEntropy( imHarAnal );
    fprintf('ENTROPY--H7 :\t%d\n',enH7);
    fprintf('ENTROPY--H6 :\t%d\n',enH6);
    fprintf('ENTROPY--H5 :\t%d\n',enH5);
    fprintf('ENTROPY--H4 :\t%d\n',enH4);
    fprintf('ENTROPY--H3 :\t%d\n',enH3);
    fprintf('ENTROPY--H2 :\t%d\n',enH2);
    fprintf('ENTROPY--H1 :\t%d\n',enH1);
    fprintf('ENTROPY--H0 :\t%d\t(whole Image)\n',enH0);
end
```

Listing 9: Η υλοποίηση του πρώτου μέρους.

```
function partA()
    fprintf('---------------PART A--------------\n');
    %% Design our 1-D input signal
    A=255;
    fx = (-A:0.1:A)';

    %% Plot 9 levels of quantization
    %keep in mind that desiredL = 2^R
    figure
    levels=9;
    fancy = ['b','k','r','m'];
    for r=0:levels-1
        fxQ = uni_scalar(fx, A/2, 2^r);
        subplot(2, 5, r+1)
        plot(fx, fxQ, fancy(mod(r,4)+1)); %choose color from fancy array
        str = sprintf('Char. Func. of uni_scalar');
        title(str,'Interpreter','none')
        str = sprintf('r=%d', r);
        legend(str,'Location',['South','East'])
        ylabel('$$\tilde{f}(x)$$', 'Interpreter', 'LaTeX');
        xlabel('$$f(x)$$', 'Interpreter', 'LaTeX');
        grid on
    end


    %% Quantize 'lena_gray_512.tif'
    figure
    lena= double( imread('lena_gray_512.tif') );

    levels = 9;
    A = max(max(lena)); % to have correct map, see partC for details
    errors = zeros(levels,1);
    for r=0:levels-1
        l = 2^r;
        q = uni_scalar(lena,A/2,l);
        %calculate mean square errors
        errors(r+1)=immse(lena,q);
        rStr = sprintf('r = %d', r);
        %print
        fprintf('MSE, %s :\t%d\n',rStr, errors(r+1));

        %illustrate the images
        subplot(2,5,r+1)
        imagesc(q)
        colormap gray; colorbar; caxis([0 255])
        title(rStr)
        grid on
    end

    %% plot rate-distortion curve
```

```
    figure;
    semilogy([0:8],errors)
    ylabel('MSE Value compared to original')
    xlabel('R Value on quantization')
    title('rate-distortion curve D(R)')
    grid on

    fprintf('----------------------------------\n');
end
```

Listing 10: Η υλοποίηση του δεύτερου μέρους.

```
function outGray = partB( input_str )
    fprintf('--------------PART B--------------\n');
    vidX=VideoReader(input_str);
    figure;
    currAxes = axes;
    numOfFr = 0;
    fps = vidX.FrameRate;
    dur = vidX.Duration;
    while hasFrame(vidX)
        vidFrame = readFrame(vidX);
        image(vidFrame, 'Parent', currAxes);
        numOfFr = numOfFr + 1;
        pause(1/fps);
    end

    info = sprintf('The total number of frames is %d\n', numOfFr);
    info = sprintf('%sThe frameRate of the video is %d\n',info, fps);
    info = sprintf('%sEach frame is %dx%d\n',info, vidX.Width, vidX.Height);
    info = sprintf('%sThe duration of the video is %d',info, dur);
    disp(info);

    vidX = VideoReader(input_str);
    inIm = read(vidX,50);
    outGray = rgb2gray(inIm);
    figure
    subplot(1,2,1); imshow(inIm); title('50^{th} frame in RGB')
    subplot(1,2,2); imshow(outGray); title('50^{th} frame in GRAY')

    fprintf('----------------------------------\n');
end
```

Listing 11: Η υλοποίηση του τρίτου μέρους.

```
function partC( inIm )
    fprintf('--------------PART C--------------\n');
    inIm = imresize(inIm,[256 256]);

    %easier to work with doubles (still display as uint8)
    inIm  =double(inIm);
    imOrig=double(inIm);

    %% 2 level Haar-2D-wavelet transformation
    imHarAnal = haar2anal( inIm, 2);

    figure
    subplot(1,2,1);
    imagesc(imHarAnal); colormap gray; colorbar;
    title('The analized Image');
    footNote = ['Note that the negative values are at most -50.' ...
        ' We later utilize that only the subbands of' ...
        ' differences have negative values.'];
    xlabel(footNote);
    axis square;
    axis equal;
    axis image;
    inIm = haar2syn( imHarAnal, 2);
```

```
    subplot(1,2,2);
    imshow(uint8(cat(2, inIm, imOrig)));
    title('left: Synthesis of HAAR, right: Original');
    xlabel('note: synthesis before quantization of subbands')

    %% For error checking, assert that decomp and comp is fine with HAAR
    errAssrt = immse(inIm,imOrig);
    assert(errAssrt == 0,'Haar anal and synth doesnt work as expected');

    %% Backup the analized image for second round of Quantization
    haarAnalBckup = imHarAnal;


    fprintf('\n\t--First Method--\n\n');
    %% Quantize all subbands but the image (H7)
    [r,c] = size(imHarAnal);
    l = 2^4; % quantize with r=4 both levels

    %since we quantize the differences in haar, our quantization window
    %must be at least [ -|min(A)|, |min(A)| ]
    A = ceil( abs( min( min(imHarAnal))));

    %H6
    imHarAnal(r/4:r/2, 1:c/4)  = uni_scalar(imHarAnal(r/4:r/2, 1:c/4)  ,A,l);
    %H5
    imHarAnal(r/4:r/2, c/4:c/2)= uni_scalar(imHarAnal(r/4:r/2, c/4:c/2),A,l);
    %H4
    imHarAnal(1:r/4  , c/4:c/2)= uni_scalar(imHarAnal(1:r/4  , c/4:c/2),A,l);
    %H3
    imHarAnal(r/2:r  , 1:c/2)  = uni_scalar(imHarAnal(r/2:r  , 1:c/2)  ,A,l);
    %H2
    imHarAnal(r/2:r  , c/2:c)  = uni_scalar(imHarAnal(r/2:r  , c/2:c)  ,A,l);
    %H1
    imHarAnal(1:r/2  , c/2:c)  = uni_scalar(imHarAnal(1:r/2  , c/2:c)  ,A,l);

    %% Entropy
    entrOfSubBands( imHarAnal );

    %% Display reConstruction
    figure;
    firstMethod = haar2syn( imHarAnal, 2);
    imshow(uint8(firstMethod));
    peakSNR = psnr(firstMethod,imOrig);
    str = sprintf('First Method Image has %0.2f dB peak-SNR', peakSNR);
    title(str);
    % compression Ratio
    originalEntropy = imEntropy(uint8(imOrig));
    compressEntropy = imEntropy(uint8(imHarAnal));
    [m, n] = size(imOrig);
    bitsOriginal = m * n * 8;
    bitsCompress = (m/4 * n/4 * 8) + 3*(m/4 * n/4 * 4) + 3*(m/2 * n/2 * 4);
    bitRatio = bitsOriginal/bitsCompress;
    ratio = originalEntropy/compressEntropy;
    fprintf('First Method: Entropy ratio is: %0.2f\n',ratio);
    fprintf('First Method: Compression ratio is: %0.2f (bit ratio)\n',bitRatio);


    fprintf('\n\t--Second Method--\n\n');
    %% Quantize first layer (H1, H2, H3) with r=3
    imHarAnal = haarAnalBckup; %restore analized image

    [r,c] = size(imHarAnal);
    l = 2^3; % r=3 for first level
    A = ceil( abs( min( min(imHarAnal))));

    %H3
    imHarAnal(r/2:r, 1:c/2)  = uni_scalar(imHarAnal(r/2:r, 1:c/2)  ,A,l);
    %H2
    imHarAnal(r/2:r, c/2:c)  = uni_scalar(imHarAnal(r/2:r, c/2:c)  ,A,l);
    %H1
    imHarAnal(1:r/2, c/2:c)  = uni_scalar(imHarAnal(1:r/2, c/2:c)  ,A,l);

```

```
    %% Quantize second layer (H4, H5, H6) with r=5
    l = 2^5; % r=5 for second level
    %H6
    imHarAnal(r/4:r/2, 1:c/4)  = uni_scalar(imHarAnal(r/4:r/2, 1:c/4)  ,A,l);
    %H5
    imHarAnal(r/4:r/2, c/4:c/2)= uni_scalar(imHarAnal(r/4:r/2, c/4:c/2),A,l);
    %H4
    imHarAnal(1:r/4  , c/4:c/2)= uni_scalar(imHarAnal(1:r/4  , c/4:c/2),A,l);

    %% Entropy
    entrOfSubBands( imHarAnal );

    %% Display reConstruction
    figure;
    secondMethod = haar2syn( imHarAnal, 2);
    imshow(uint8(secondMethod));
    peakSNR = psnr(secondMethod,imOrig);
    str = sprintf('Second Method Image has %0.2f dB peak-SNR', peakSNR);
    t = title(str);
    % compression Ratio
    originalEntropy = imEntropy( uint8(imOrig) );
    compressEntropy = imEntropy( uint8(imHarAnal) );
    ratio = originalEntropy/compressEntropy;
    [m, n] = size(imOrig);
    bitsOriginal = m * n * 8;
    bitsCompress = (m/4 * n/4 * 8) + 3*(m/4 * n/4 * 5) + 3*(m/2 * n/2 * 3);
    bitRatio = bitsOriginal/bitsCompress;
    fprintf('Second Method: Entropy ratio is: %0.2f\n',ratio);
    fprintf('Second Method: Compression ratio is: %0.2f (bit ratio)\n',bitRatio);

    fprintf('---------------------------------\n');
end
```