Συστήματα Πολυμέσων
και
Εικονική Πραγματικότητα

Εργασία 2019 – 2020

Αναφορά Εργασίας

Εμμανουήλ Χρήστος 8804

Πέμπτη, 9 Ιανουαρίου 2020

Περιεχόμενα

Εισαγωγή

Στα πλαίσια του μαθήματος Συστήματα Πολυμέσων και Εικονική Πραγματικότητα, ζητήθηκε η υλοποίηση ενός κωδικοποιητή/αποκωδικοποιητή ακίνητης εικόνας. Ο κωδικοποιητής/αποκωδικοποιητής πρέπει να συμβαδίζει όσο το περισσότερο γίνεται με το πρότυπο JPEG (ISO/IEC 10918 - 1:1994) και συγκεκριμένα την εκδοχή seline sequential DCT – based.

Η εκπόνηση της εργασίας πραγματοποιήθηκε στο πρόγραμμα MATLAB. Με την βοήθεια των σημειώσεων του μαθήματος, τις οδηγίες του προτύπου, την εκφώνηση της εργασίας καθώς και μια γκάμα υλικού βοηθητικού περιεχομένου επιτεύχθηκε η υλοποίηση μιας σειράς συναρτήσεων που καταφέρνουν να απαντήξουν τα ερωτήματα που τέθηκαν και συντελούν στην δημιουργία ενός κωδικοποιητή/αποκωδικοποιητή.

Η εργασία, όπως δίνεται από την εκφώνηση, χωρίζεται σε 3 μέρη :

- JPEG Library
- JPEG Intergration
- JPEG Syntax

Στην συγκεκριμένη υλοποίηση επιτεύχθηκε η ολοκλήρωση των δύο πρώτων μερών.

Η αναφορά αυτή έχει σκοπό την επεξήγηση των επιμέρους μερών και ανάλυση σημείων που δεν καλύπτονται από την εκφώνηση της εργασίας. Επιπλέον στοχεύει στην περιγραφή του τρόπου λειτουργίας, απαντάει στα ζητούμενα ερωτήματα και παρουσιάζει παραδείγματα.

JPEG Library

Στο πρώτο μέρος ζητείται η κατασκευή μιας βιβλιοθήκης συναρτήσεων. Κάθε συνάρτηση έχει και μία αντίστροφη. Με την σύνθεση των συναρτήσεων αυτών με συγκεκριμένο τρόπο επιτυγχάνεται η δημιουργία ενός κωδικοποιητή και ενός αποκωδικοποιητή.

Για τον λόγο αυτό κάθε συνάρτηση με την αντιστροφή της πραγματοποιούν μία από της λειτουργίες του συνόλου λειτουργιών που αποσκοπεί στην συμπίεση και κωδικοποίηση (αποκωδικοποίηση και αποσυμπιέσει) μιας ακίνητης εικόνας.

Το σύνολο των λειτουργιών είναι :

- Προεπεξεργασία
- Μετασχηματισμός DCT
- Κβαντισμός
- ZigZag scanning & RLE
- Κωδικοποίηση Huffman

Στο τέλος του 1ου μέρους, μετά την υλοποίηση αυτών των πέντε συναρτήσεων και των αντιστροφών τους, ζητείται η δημιουργία ενός demo που εξετάζει την λειτουργία μέρους των συναρτήσεων και οπτικοποιεί τα αποτελέσματα.

Προεπεξεργασία

convert2ycbcr

Η συνάρτηση που κατασκευάζεται είναι η :

```
function [imageY, imageCb, imageCr] = convert2ycbcr(imageRGB, subimg)
```

Οι ενέργειες που εκτελεί είναι αρχικά η διόρθωση των διαστάσεων της εικόνας imageRGB σε πολλαπλάσιο του 8, στην συνέχεια η μετατροπή των πινάκων R G B σε Y Cb Cr και τέλος η εφαρμογή της υποδειγματολιψίας του δίνεται από το subimg.

Περαιτέρω εξηγήσεις δίνονται για το σημείο της μετατροπής. Αρχικά η εικόνα εισάγεται σε μορφή τρισδιάστατου πίνακα ακέραιων στο διάστημα [0 , 255]. Με σκοπό την συντήρηση της πληροφορίας γίνετε αρχικά η μετατροπή των ακεραίων σε δεκαδικούς στο διάστημα [0, 1] και στην συνέχεια διασπάτε ο τρισδιάστατος πινάκας σε 3 δυσδιάστατους πίνακες.

Οι τρις πίνακες είναι οι συνιστώσες R G B, που με τους παρακάτω τύπους μετατρέπονται σε συνιστώσες Y Cb Cr :

Conversion to $y'c_Bc_R$ from r'g'b'			
y' =	0.299 * r' +	0.587 * g' +	0.114 * b'
c_B =	-0.168736 * r' +	-0.331264 * g' +	0.500 * b'
c_R =	0.500 * r' +	-0.418688 * g' +	-0.081312 * b'

Ο πίνακας Y περιέχει δεκαδικούς στο διάστημα [0, 1], ενώ οι Cb και Cr δεκαδικούς στο διάστημα [-0.5, 0.5]. Με κατάλληλες πράξεις μετατρέπονται οι 3 πίνακες σε πίνακες δεκαδικών που περιέχουν τιμές στο διάστημα [0, 255], οι πίνακες αυτοί θα υποστούν την υπερδειγματολιψία.

convert2rgb

Η αντίστροφη συνάρτηση που κατασκευάζεται είναι η :

```
function imageRGB = convert2rgb(imageY, imageCr, imageCb, subimg)
```

Ο ρόλος της συνάρτησης αυτής είναι να δεχτεί 3 πίνακες, τους Y Cb Cr, να καταλάβει την υπερδειγματολιψία που έχει γίνει, να ανακατασκευάσει την χαμένη πληροφορία και τελικά να δημιουργήσει τον τελικό τρισδιάστατο πίνακα RGB, που περιέχει ακέραιους στο διάστημα [0, 255].

Η αναγνώριση της υφιστάμενης υποδειγματολιψίας γίνετε με σύγκριση του μεγέθους του πίνακα Y με τα μεγέθη των πινάκων Cb και Cr.

Η ανακατασκευή της χαμένης πληροφορίας γίνεται με τον υπολογισμό του μέσου όρου των διπλανών κελίων. Αρχικά με χρήση της συνάρτησης **transpose** προστίθενται μηδενικά σε κατάλληλες γραμμές ή και στήλες (ανάλογα την υποδειγματολιψίας) στους πίνακες Cb και Cr και στην συνέχεια αντικαθίστανται τα μηδενικά με τον μέσο όρο των δύο διπλανών κελιών.

Στην συνέχεια γίνεται η μετατροπή στις συνιστώσες R G B και η δημιουργία του τρισδιάστατου πίνακα. Οι 3 πίνακες είναι πίνακες δεκαδικών στο διάστημα [0, 255], με κατάλληλες πράξεις και με τους παρακάτω τύπους οι πίνακες R G B που προκύπτουν περιέχουν ακέραιους στο διάστημα [0, 255].

Conversion to r'g'b' from $y'c_Bc_R$			
$r' = 1.0 * y' +$	$0 * c_B +$	$1.402 * c_R$
$g' = 1.0 * y' + -0.344136 * c_B + -0.714136 * c_R$		
$b' = 1.0 * y' +$	$1.772 * c_B +$	$0 * c_R$

Μετασχηματισμός DCT

blockDCT

Ρόλος του συγκεκριμένου βήματος είναι η αποσυσχέτιση των δεδομένων. Η συνάρτηση που αναλαμβάνει αυτό τον ρόλο είναι η παρακάτω :

```
function dctBlock = blockDCT(block)
```

Η συνάρτηση αυτή δέχεται block μεγέθους 8x8 και αφού εφαρμόσει τον μετασχηματισμό DCT με την συνάρτηση **dct2** του MATLAB, παράγει το μετασχηματισμένο block στην έξοδο.

Αναφέρετε ότι τα μπλοκ στην είσοδο έχουν δεκαδικές τιμές στο διάστημα [0, 255]. Για λόγους καλύτερης ποιότητας ανακατασκευασμένης εικόνας, πριν τον μετασχηματισμό, οι τιμές μετατοπίζονται στο διάστημα [-127.5, 127.5].

iBlockDCT

Η αντίστροφη της προηγούμενης συνάρτησης είναι η :

```
function block = iBlockDCT(dctBlock)
```

Δέχεται ένα μετασχηματισμένο μπλοκ 8x8 και παράγει το αρχικό μπλοκ. Χρησιμοποιείται η συνάρτηση **idct2** του MATLAB και μετά τον μετασχηματισμό μετατοπίζονται οι δεκαδικές τιμές του αρχικού μπλοκ στο διάστημα [0, 255].

Κβαντισμός

quantizeJPEG

Στην συνέχεια της διαδικασίας είναι απαραίτητη μια συνάρτηση κβαντισμού. Οι τιμές που υπάρχουν στα μετασχηματισμένα μπλοκ πρέπει να αντιστοιχιστούν σε συγκεκριμένες στάθμες. Η συνάρτηση που υλοποιείται είναι η :

```
function qBlock = quantizeJPEG(dctBlock, qTable, qScale)
```

Δέχεται ένα μετασχηματισμένο μπλοκ, έναν πίνακα 8x8 (ο πίνακας θα δοθεί στο 2^{ο} μέρος της εργασίας) και έναν αριθμό. Ο αριθμός αυτός πολλαπλασιάζεται με τον πίνακα και δημιουργούν έτσι έναν τελικό πίνακα 8x8, τα στοιχεία του οποίου διαιρούν τα στοιχεία των αντίστοιχων θέσεων του μπλοκ. Στρογγυλοποιώντας τα αποτελέσματα των διαιρέσεων αντιστοιχίζονται οι αριθμοί του μπλοκ στις στάθμες και στην έξοδο δημιουργείται το κβαντισμένο πλέον μπλοκ.

dequantizeJPEG

Η αντίστροφη συνάρτηση καλείται να πολλαπλασιάσει το κβαντισμένο μπλοκ με τον ίδιο πίνακα, προσπαθώντας να κατασκευάσει το αρχικό μετασχηματισμένο μπλοκ που χρησιμοποιήθηκε στον κβαντισμό. Η διαδικασία αυτή εμπεριέχει σημαντική απώλεια πληροφορίας καθώς δεν είναι δυνατή η τέλεια ανακατασκευή του μετασχηματισμένου μπλοκ.

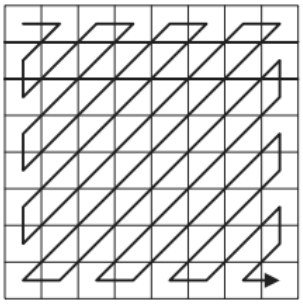
Η συνάρτηση που δημιουργήθηκε φαίνεται παρακάτω :

```
function dctBlock = dequantizeJPEG(qBlock, qTable, qScale)
```

ZigZag scanning & RLE

Στο σημείο αυτό γίνεται το λεγόμενο Zig-Zag scanning και εκτελείται ο αλγόριθμος Run Length.

Το **Zig-Zag scanning** είναι μια διαδικασία αποτύπωσης των στοιχείων ενός κβαντισμένου μπλοκ σε ένα μονοδιάστατο πίνακα με την παρακάτω σειρά :



Ο αλγόριθμος **Run Length** μετατρέπει τον μονοδιάστατο πίνακα, που προκύπτει από το Zig Zag, σε έναν πίνακα 2 στηλών και R γραμμών. Στην δεύτερη στήλη αποθηκεύεται ο μη μηδενικός αριθμός που βρέθηκε στον μονοδιάστατο πίνακα, ενώ στην πρώτη ο αριθμός των μηδενικών που προσπελάστηκαν μέχρι την εύρεση του μη μηδενικού αριθμού (υπάρχουν εξαιρέσεις σε αυτή την διαδικασία οι οποίες θα αναφερθούν στην συνέχεια).

runSymbols

Η συνάρτηση που υλοποιείται στο MATLAB είναι η :

```
function runSymbols = runLength(qBlock, DCpred)
```

Αξίζει να αναφερθεί η διαδικασία και να τονιστούν ορισμένα, κρίσιμα για την συνολική λειτουργία ενός κωδικοποιητή σημεία.

Η διαδικασία έχει ως εξής :

1. Μετασχηματισμός του `qBlock` σε ένα διάνυσμα (ZZ) μέσο ρουτίνας που υλοποιεί το zig-zag scanning.
2. Το πρώτο στοιχείο ενός block και συνεπώς του ZZ αποτελεί την DC συνιστώσα. Η DC συνιστώσα έχει διαφορετικό τρόπο αποθηκεύσεις από τον υπόλοιπο πίνακα (AC συνιστώσες) και για αυτό από το 1^{ο} στοιχείο του ZZ αφαιρείται το `DCpred`.
3. Για λόγους που θα φανούν στην συνέχεια της διαδικασίας πρέπει το εύρος της DC συνιστώσας να περιορίζεται στο διάστημα [-2047, 2047], ενώ των AC συνιστωσών στο διάστημα [-1023, 1023].
4. Εκτελείται η ρουτίνα του Run Length αλγόριθμου, λαμβάνοντας υπόψιν 2 σημαντικές εξαιρέσεις. Πρώτον για μήκος 16 μηδενικών στην πρώτη στήλη του `runSymbols` αποθηκεύεται το 15 και στην δεύτερη το 0 (**ZRL**). Δεύτερον αν βρεθεί στο τέλος του πίνακα και υπάρχει μηδενικό αποθηκεύεται και στις 2 στήλες το 0 (**EOB**). Αυτές οι δύο περιπτώσεις καθορίζονται από την κωδικοποίηση Huffman και θα γίνει κατανοητός ο ρόλος τους αργότερα.
5. Τέλος πρέπει να γίνει η σωστή τοποθέτηση του τέλος του μπλοκ. Τέλος του μπλοκ θεωρείται το σημείο μετά το τελευταίο μη μηδενικό στοιχείο του πίνακα. Έτσι σε περίπτωση που στον πίνακα `runSymbols` υπάρχουν συνεχόμενοι όροι [15, 0] και στην συνέχεια το [0, 0], πρέπει να γίνει απαλοιφή των πλεονάζοντων όρων και να μετακινηθεί το [0, 0] πιο πάνω.

irunLength

Η αντιστροφή συνάρτηση είναι η :

```
function qBlock = irunLength(runSymbols, DCpred)
```

Ρόλος της συνάρτησης αυτής είναι να δεχτεί έναν πίνακα `runSymbols` και να επιστρέψει ένα κβαντισμένο μπλοκ.

Αρχικά χρησιμοποιώντας το πίνακα `runSymbols` φτιάχνει το διάνυσμα ZZ, προσθέτει στο πρώτο στοιχείο το `DCpred` και τέλος χρησιμοποιώντας την αντίστροφη ρουτίνα του zig-zag δημιουργεί ένα μπλοκ 8x8 που περιέχει της κβαντισμένες τιμές των DCT συντελεστών.

Κωδικοποίηση Huffman

Η κωδικοποίηση Huffman είναι η διαδικασία κατά την οποία τα σύμβολα του πίνακα `runSymbols` (κάθε γραμμής αποτελεί ένα σύμβολο) μεταφράζονται σε δυαδικές λέξεις, δηλαδή αναπαραστάσεις 0 και 1.

Η κωδικοποίηση πρέπει να γίνεται με τέτοιο τρόπο, ώστε κάθε λέξη ενός συμβόλου να μην είναι πρόθεμα άλλου συμβόλου. Ένα παράδειγμα του κωδικοποιημένου συμβόλου **[0, 7]** είναι :

Δυαδική Λέξη	**Δυαδική Τιμή**
1 0 0	1 1 1

Η κωδικοποίηση αποτελείται από δύο μέλη. Το δεύτερο μέλος είναι η δυαδική αναπαράσταση του αριθμού της δεύτερης στήλης, στο παράδειγμα το 7 στο δυαδικό σύστημα είναι το 111 και απαιτείται ο ελάχιστος αριθμός τριών bits. Το πρώτο μέλος είναι η δυαδική λέξη και εξαρτάται τόσο από το πλήθος των προσπελάσιμων μηδενικών (αριθμός πρώτης στήλης) όσο και από το ελάχιστο αριθμό bits που απαιτούνται για την δυαδική αναπαράσταση.

Στην υλοποίηση την συγκεκριμένης εργασίας δεν υλοποιείται ο καθορισμός των μη-προθεματικών δυαδικών λέξεων. Αντιθέτως οι συναρτήσεις που κατασκευάστηκαν χρησιμοποιούν έτοιμους πίνακες δυαδικών λέξεων, που δίνονται από το πρότυπο και εντοιχίζουν κάθε σύμβολο σε μια λέξη του πίνακα. Οι πίνακες αυτοί θα δοθούν στο δεύτερο κομμάτι της εργασίας.

huffEnc

Η συνάρτηση που αναλαμβάνει την αντιστοίχιση των συμβόλων στις κατάλληλες λέξεις και συνεπώς την κωδικοποίηση τους, είναι η παρακάτω :

```
function huffStream = huffEnc(runSymbols)
```

Το `huffStream` που παράγεται στο τέλος της διαδικασίας είναι ένα stream από 0 και 1 και αποτελεί την κωδικοποιημένη αναπαράσταση ενός μπλοκ.

Η συνάρτηση δέχεται των πίνακα συμβόλων ενός μπλοκ, η κωδικοποίηση της DC συνιστώσας ($1^{η}$ γραμμή) διαφέρει από τις υπόλοιπες γραμμές του πίνακα, συγκεκριμένα απαιτούνται διαφορετική πίνακες δυαδικών λέξεων.

Εν συντομία η διαδικασία κωδικοποίησης κάθε συμβόλου είναι :

1. Έλεγχος αρνητικού πρόσημου.
2. Υπολογισμός ελάχιστου αριθμού bits που απαιτείται από την διάδικοι αναπαράσταση.
3. Αντιστοίχιση σε μια δυαδική λέξη ενός πίνακα Huffman, σύμφωνα με τον αριθμού bits και το πλήθος προσπελασμένων μηδενικών (το πλήθος προσπελασμένων μηδενικών αφορά τις AC συνιστώσες).
4. Μετατροπή της απόλυτης τιμής του αριθμού στον αντίστοιχο δυαδικό.
5. Αν ο αριθμός ήταν αρνητικός, υπολογίζεται η συμπληρωματική δυαδική αναπαράσταση έτσι ώστε το MSB να είναι 0. (Αφού χρησιμοποιείται το ελάχιστο μέγεθος δυαδικής αναπαράστασης για έναν θετικό αριθμό το MSB θα είναι πάντα 1. Έτσι ο παραπάνω τρόπος διευκολύνει έναν αποκωδικοποιητή να καταλάβει από την τιμή του MSB αν η δυαδική τιμή που θα διαβαστεί αφορά αρνητικό ή θετικό αριθμό).
6. Ένωση της δυαδικής λέξης και της δυαδικής αναπαράστασης όπως φαίνεται και στο παραπάνω παράδειγμα.

Στο τέλος της διαδικασίας τοποθετούνται οι κωδικοποιημένες αναπαραστάσεις των συμβόλων η μία μετά την άλλη, παράγοντας έτσι το τελικό bits stream του block.

huffDec

Η συνάρτηση αυτή έχει τον αντίστροφο ρόλο, πρέπει να διαβάζει τις κωδικοποιημένες αναπαραστάσεις των μπλοκς και να παράγει σωστούς πίνακες συμβόλων.

```
function runSymbols = huffDec(huffStream)
```

Απαιτούνται και εδώ οι ίδιοι πίνακες , διαφορετικός πίνακας για την DC συνιστώσα και διαφορετικός για τις AC συνιστώσες.

Για την επίτευξη του στόχου της συνάρτησης, χρησιμοποιούνται 3 δείκτες. Ο πρώτος δείκτης χρησιμοποιείτε για το γέμισμα του τελικού πίνακα συμβόλων. Οι άλλοι δύο ανατρέχουν το stream, μεταβάλουν την απόσταση μεταξύ τους προσπαθώντας να εντοπίσουν κομμάτια τους stream που θα ταυτιστούν με μια δυαδική λέξη του πίνακα και στην συνέχεια μετατοπίζονται κατάλληλα για να διαβάσουν την δυαδική τιμή. Η διαδικασία επαναλαμβάνεται μέχρι το τέλος του stream.

Demo 1

Στο τέλος του πρώτου μέρους της εργασίας ζητήθηκε η δημιουργία ενός script (**demo1.m**), το οποίο εξετάζει την λειτουργία μερικών συναρτήσεων και οπτικοποιεί τα αποτελέσματα.

Ερώτημα Α

Το πρώτο ερώτημα εξετάζει την λειτουργία των συναρτήσεων :

- `convert2ycbcr`
- `convert2rgb`

Ζητείται ο μετασχηματισμός δύο εικόνων από RGB σε YCbCr και πάλι σε RGB :

- Εικόνα 1 : υποδειγματοληψία 4:2:2
- Εικόνα 2 : υποδειγματοληψία 4:4:4

Παρακάτω παρουσιάζονται τα αποτελέσματα :

Image 1 - Original

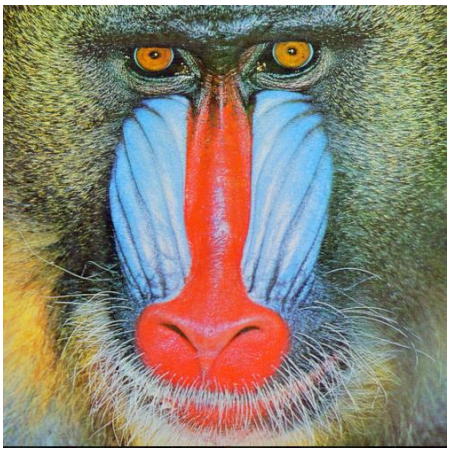


Image 1 - Reconstructed (Downsample 4:2:2)

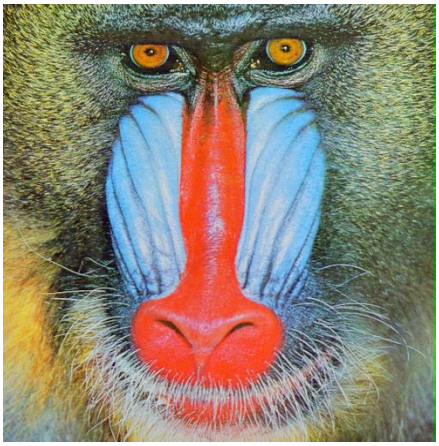


Image 2 - Original

Image 2 - Reconstructed (Downsample 4:4:4)

Ερώτημα Β

Το δύετερο ερώτημα εξετάζει την λειτουργία των συναρτήσεων :

- `convert2ycbcr`
- `blockDCT`
- `quantizeJPEG`
- `dequantizeJPEG`
- `iBlockDCT`
- `convert2rgb`

Ζητείται ο μετασχηματισμός δύο εικόνων από RGB σε κβαντισμένους συντελεστές DCT και πάλι σε RGB :

- Εικόνα 1 : υποδειγματοληψία 4:2:2 | qScale 0.6
- Εικόνα 2 : υποδειγματοληψία 4:4:4 | qScale 5.0

Παρακάτω παρουσιάζονται τα αποτελέσματα :

Image 1 - Original

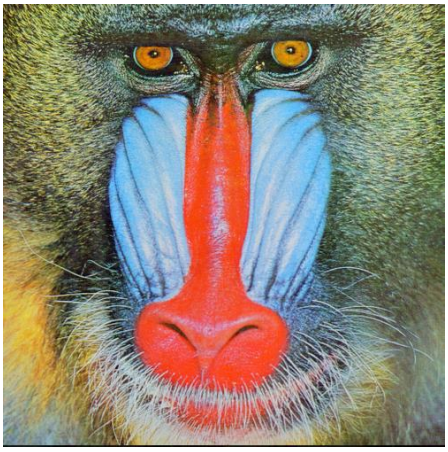


Image 1 – Reconstructed (Downsample 4:2:2 | qScale = 0.6)

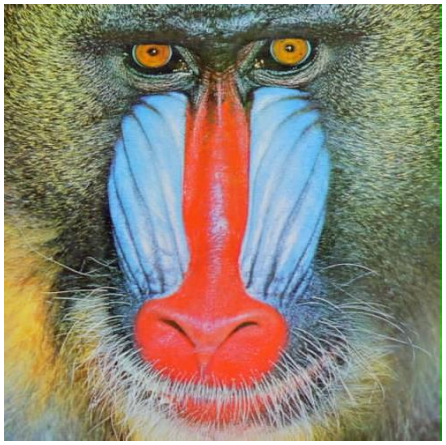
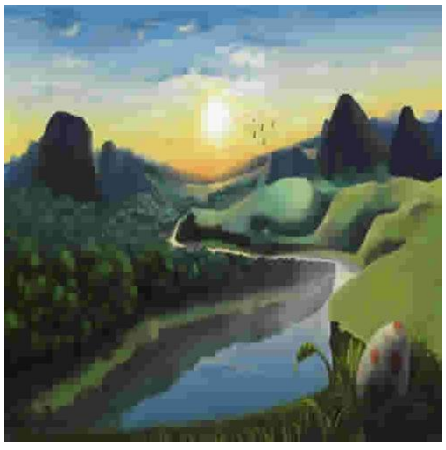


Image 2 - Original

Image 2 – Reconstructed (Downsample 4:4:4 | qScale = 5)



JPEG Integration

Στο δεύτερο κομμάτι της εργασίας επιτυγχάνεται ο συνδυασμός των παραπάνω συναρτήσεων με σκοπό την δημιουργία μιας πρώιμης μορφής κωδικοποιητή και αποκωδικοποιητή.

Ο κωδικοποιητής κατασκευάζεται με την χρήση των πέντε κύριων συναρτήσεων που αναλύθηκαν στο πρώτο μέρος της αναφοράς :

1. convert2ycbcr
2. blockDCT
3. quantizeJPEG
4. runLength
5. huffEnc

Ενώ η κατασκευή του αποκωδικοποιητή στηρίζετε στις πέντε αντίστροφες αυτών συναρτήσεις :

1. huffDec
2. irunLength
3. dequantizeJPEG
4. iBlockDCT
5. convert2rgb

Στο τέλος του δεύτερου μέρους παρουσιάζονται συγκεκριμένες μετρήσεις που αφορούν την ποιότητα της διαδικασίας κωδικοποίησης και αποκωδικοποίησης.

JPEG Encoder

Στο σημείο αυτό δημιουργείται η συνάρτηση που αναλαμβάνει τον ρόλο του κωδικοποιητή. Αποτελεί την συνάρτηση που δέχεται μια εικόνα και παράγει την κωδικοποίηση της.

```
function JPEGenc = JPEGencode(img, subimg, qScale)
```

Η παραπάνω συνάρτηση δέχεται μια εικόνα , τον τύπο υποδειγματολιψίας καθώς και τον συντελεστή κβαντισμού, όσο μεγαλώνει το qScale αυξάνεται τόσο η συμπίεση όσο και η απώλεια πληροφορίας της εικόνας.

Μετά την διεκπεραίωση συγκεκριμένης διαδικασίας παράγεται το τελικό cell από struct σύμφωνα με τις οδηγίες της εκφώνησης της εργασίας.

Στο σώμα της συγκεκριμένης συνάρτησης γίνεται και η αρχικοποίηση των πινάκων κβαντισμού και πινάκων δυαδικών λέξεων Huffman, οι οποίοι σημειώνεται ότι αποθηκεύονται σαν cells που σε κάθε θέση τους περιέχουν πίνακες χαρακτήρων. Κάθε πίνακας χαρακτήρων αποτελεί μια δυαδική λέξη. Σύμφωνα με το πρότυπο η διαδικασία χρησιμοποιεί τις συνιστώσες Y Cb Cr. Απαιτούνται διαφορετικοί πίνακες για των κβαντισμό και την κωδικοποίηση της συνιστώσας της φωτεινότητας (Luminance Y) και διαφορετικοί για τις χρωματικές συνιστώσες (Chrominance Cb, Cr). Έτσι είναι αναγκαία η αρχικοποίηση όλων των απαραίτητων πινάκων.

Πίνακες Κβαντισμού :

Luminance

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Chrominance

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Παρατηρείτε αύξηση των τιμών κατά την κίνηση προς τα δεξιά και κάτω. Αυτό σημαίνει ότι οι συντελεστές DCT εκείνων των σημείων είναι πιθανότερο να αντιστοιχιστούν στην στάθμη 0 μετά την διαίρεση με έναν από τους παραπάνω πίνακες (Κβαντισμός). Αυτοί η συντελεστές όμως αναφέρονται σε υψηλές συχνότητες – σε αλλαγές της εικόνας που συμβαίνουν σε πολύ μικρές αποστάσεις και αρκετές φορές δεν επηρεάζουν το πώς ένας παρατηρητής αντιλαμβάνεται μια εικόνα. Έτσι η απαλοιφή αυτών των συντελεστών καταφέρνει να ελαττώσει το μέγεθος της πληροφορίας που θα κωδικοποιηθεί και ταυτόχρονα η ανακατασκευασμένη εικόνα να φαίνεται ίδια από κάποιον παρατηρητή.

Πίνακες Huffman - Luminance :

DC component

	1	2	3	4	5	6	7	8	9	10	11	12
1	00	010	011	100	101	110	1110	11110	111110	1111110	11111110	111111110

AC components

	1	2	3	4	5	6	7	8	9	10	11
1	1010	00	01	100	1011	11010	1111000	11111000	1111110110	1111111110000010	1111111110000011
2		1100	11011	1111001	111110110	11111110110	1111111110000100	1111111110000101	1111111110000110	1111111110000111	1111111110001000
3		11100	11111001	1111110111	111111110100	1111111110001001	1111111110001010	1111111110001011	1111111110001100	1111111110001101	1111111110001110
4		111010	111110111	111111110101	1111111110001111	1111111110010000	1111111110010001	1111111110010010	1111111110010011	1111111110010100	1111111110010101
5		111011	1111111000	1111111110010110	1111111110010111	1111111110011000	1111111110011001	1111111110011010	1111111110011011	1111111110011100	1111111110011101
6		1111010	111111101111	1111111110011110	1111111110011111	1111111110100000	1111111110100001	1111111110100010	1111111110100011	1111111110100100	1111111110100101
7		1111011	111111110110	1111111110100110	1111111110100111	1111111110101000	1111111110101001	1111111110101010	1111111110101011	1111111110101100	1111111110101101
8		11111010	111111110111	1111111110101110	1111111110101111	1111111110110000	1111111110110001	1111111110110010	1111111110110011	1111111110110100	1111111110110101
9		111111000	1111111110000000	1111111110110110	1111111110110111	1111111110111000	1111111110111001	1111111110111010	1111111110111011	1111111110111100	1111111110111101
10		111111001	1111111110111110	1111111110111111	1111111111000000	1111111111000001	1111111111000010	1111111111000011	1111111111000100	1111111111000101	1111111111000110
11		111111010	1111111111000111	1111111111001000	1111111111001001	1111111111001010	1111111111001011	1111111111001100	1111111111001101	1111111111001110	1111111111001111
12		1111111001	1111111111010000	1111111111010001	1111111111010010	1111111111010011	1111111111010100	1111111111010101	1111111111010110	1111111111010111	1111111111011000
13		1111111010	1111111111011001	1111111111011010	1111111111011011	1111111111011100	1111111111011101	1111111111011110	1111111111011111	1111111111100000	1111111111100001
14		11111111000	1111111111100010	1111111111100011	1111111111100100	1111111111100101	1111111111100110	1111111111100111	1111111111101000	1111111111101001	1111111111101010
15		1111111111101011	1111111111101100	1111111111101101	1111111111101110	1111111111101111	1111111111110000	1111111111110001	1111111111110010	1111111111110011	1111111111110100
16	11111111001	1111111111110101	1111111111110110	1111111111110111	1111111111111000	1111111111111001	1111111111111010	1111111111111011	1111111111111100	1111111111111101	1111111111111110

Πίνακες Huffman - Chrominance :

DC component

	1	2	3	4	5	6	7	8	9	10	11	12
1	00	01	10	110	1110	11110	111110	1111110	11111110	111111110	1111111110	11111111110

AC components

	1	2	3	4	5	6	7	8	9	10	11
1	00	01	100	1010	11000	11001	111000	1111000	111110100	1111110110	111111110100
2		1011	111001	11110110	111110101	11111110110	1111111110101	1111111110001000	1111111110001001	1111111110001010	1111111110001011
3		11010	11110111	1111110111	111111110110	1111111110000010	1111111110001100	1111111110001101	1111111110001110	1111111110001111	1111111110010000
4		11011	11111000	1111111000	111111110111	1111111110010001	1111111110010010	1111111110010011	1111111110010100	1111111110010101	1111111110010110
5		111010	111110110	1111111110010111	1111111110011000	1111111110011001	1111111110011010	1111111110011011	1111111110011100	1111111110011101	1111111110011110
6		111011	1111111001	1111111110011111	1111111110100000	1111111110100001	1111111110100010	1111111110100011	1111111110100100	1111111110100101	1111111110100110
7		1111001	11111110111	1111111110100111	1111111110101000	1111111110101001	1111111110101010	1111111110101011	1111111110101100	1111111110101101	1111111110101110
8		1111010	11111111000	1111111110101111	1111111110110000	1111111110110001	1111111110110010	1111111110110011	1111111110110100	1111111110110101	1111111110110110
9		11111001	1111111110110111	1111111110111000	1111111110111001	1111111110111010	1111111110111011	1111111110111100	1111111110111101	1111111110111110	1111111110111111
10		1111110111	1111111111000000	1111111111000001	1111111111000010	1111111111000011	1111111111000100	1111111111000101	1111111111000110	1111111111000111	1111111111001000
11		111111000	1111111111001001	1111111111001010	1111111111001011	1111111111001100	1111111111001101	1111111111001110	1111111111001111	1111111111010000	1111111111010001
12		111111001	1111111111010010	1111111111010011	1111111111010100	1111111111010101	1111111111010110	1111111111010111	1111111111011000	1111111111011001	1111111111011010
13		111111010	1111111111011011	1111111111011100	1111111111011101	1111111111011110	1111111111011111	1111111111100000	1111111111100001	1111111111100010	1111111111100011
14		11111111001	1111111111100100	1111111111100101	1111111111100110	1111111111100111	1111111111101000	1111111111101001	1111111111101010	1111111111101011	1111111111101100
15		1111111111000000	1111111111101101	1111111111101110	1111111111101111	1111111111110000	1111111111110001	1111111111110010	1111111111110011	1111111111110100	1111111111110101
16	1111111010	1111111111000011	1111111111110110	1111111111110111	1111111111111000	1111111111111001	1111111111111010	1111111111111011	1111111111111100	1111111111111101	1111111111111110

Η αναπαράσταση των πινάκων φαίνεται καλύτερα στο αρχείο κειμένου του προτύπου καθώς και στο αρχείο που περιλαμβάνει την υλοποίηση της συγκεκριμένης συνάρτησης.

Οι πίνακες DC συνιστωσών είναι μονοδιάστατοι καθώς η κατηγορία που θα επιλεχθεί εξαρτάται αποκλειστικά από το πλήθος των απαιτουμένων bits της δυαδικής τιμής. Ενώ οι πίνακες των AC συνιστωσών εξαρτώνται από το μέγεθος των απαιτουμένων bits της δυαδικής τιμής (στήλες) όσο και από των αριθμό των μηδενικών που προσπελάστηκαν (γραμμές).

Έτσι αφού γίνει η αρχικοποίηση των παραπάνω πινάκων και η αποθήκευση τους στην πρώτη θέση του cell `JPEGenc` σύμφωνα με την εκφώνηση, ακολουθεί η κύρια διαδικασία κωδικοποίησης μιας εικόνας.

Μια σύντομη περιγραφή των βημάτων που ακολουθούνται είναι :

1. Υπολογισμός Y Cb Cr συνιστωσών με την `convert2ycbcr.`

2. Αντιστοίχηση πινάκων κβαντισμού και Huffman σε αυτούς που αφορούν την συνιστώσα φωτεινότητας Y.

3. Διαχωρισμός του δισδιάστατου πίνακα της συνιστώσας σε μπλοκ 8x8.

4. Μετασχηματισμός κάθε μπλοκ σύμφωνα με τον DCT μετασχηματισμό και την συνάρτηση `blockDCT.`

5. Κβαντισμός κάθε μπλοκ με την `quantizeJPEG.`

6. Δημιουργία συμβόλων του μπλοκ με την `runLength.`

7. Κωδικοποίηση των συμβόλων με την `huffEnc.`

8. Προσθήκη πληροφοριών κάθε μπλοκ στην αντίστοιχη θέση του cell `JPEGenc` σύμφωνα με την εκφώνηση.

9. Επανάληψη των βημάτων 2 – 8 για της χρωματικές συνιστώσες Cb και Cr.

Με την παραπάνω διαδικασία επιτυγχάνεται η δημιουργία του `JPEGenc.`

JPEG Decoder

Η δημιουργία του αποκωδικοποιητή γίνεται με την συνάρτηση

```
function imgRec = JPEGdecode(JPEGenc)
```

Αποτελεί αντίστροφη διαδικασία της κωδικοποίησης, μια σύντομη περιγραφή είναι :

- Υπολογισμός απαραίτητων πινάκων κβαντισμού και κωδικοποίησης από το πρώτο struct του cell `JPEGenc`.

- Για τα επόμενα struct του `JPEGenc` ακολουθείτε η παρακάτω επαναληπτική διαδικασία :

    1. Καθορισμός κατηγορίας του struct με την βοήθεια της σταθεράς `blkType`.

    2. Καθορισμός πινάκων ανάλογα με την κατηγορίας.

    3. Αποκωδικοποίηση και δημιουργία συμβόλων επιθυμητού μπλοκ με την `huffDec`.

    4. Δημιουργία κβαντισμένου μπλοκ DCT συντελεστών με την `irunLength`.

    5. Δημιουργία DCT μπλοκ με την `dequantizeJPEG`.

    6. Υπολογισμός αρχικού μπλοκ με την `iBlockDCT`.

    7. Προσθήκη του μπλοκ στην κατάλληλη θέση του δυσδιάστατο πίνακα της κατάλληλης συνιστώσας με την βοήθεια των `indHor`, `indVer`.

- Δημιουργία ανακατασκευασμένης εικόνας `imgRec` με την `convert2rgb`.

Καταγραφή Αποτελεσμάτων

qScale

Στα πλαίσια της εργασίας ζητήθηκε να μελετηθεί η επιρροή του qScale στην ποιότητα της ανακατασκευασμένης εικόνας. Για τον λόγο αυτό για διάφορες τιμές του qScale μετρήθηκε το μέσο τετραγωνικό σφάλμα και το πλήθος των bits.

Οι μετρήσεις έγιναν για δύο εικόνες με την βοήθεια του αρχείου **qScaleInfluence.m** και τα αποτελέσματα παρουσιάζονται παρακάτω :

Image 1

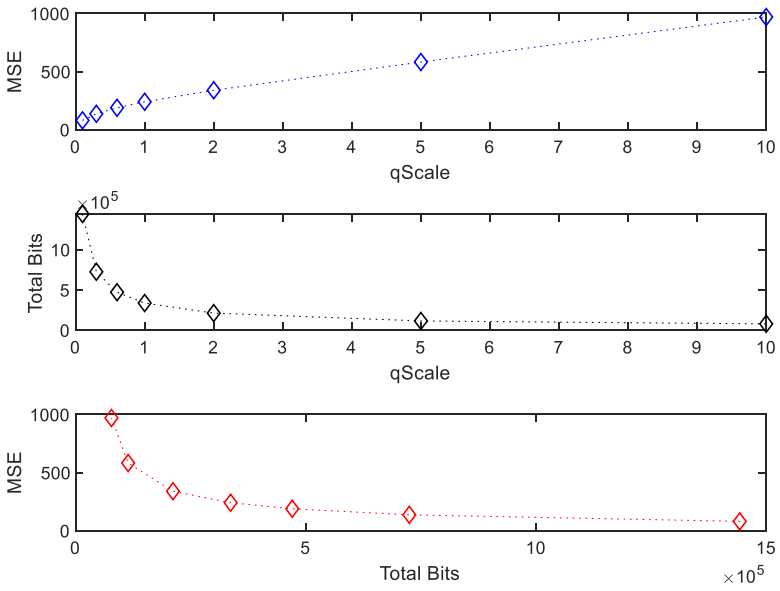
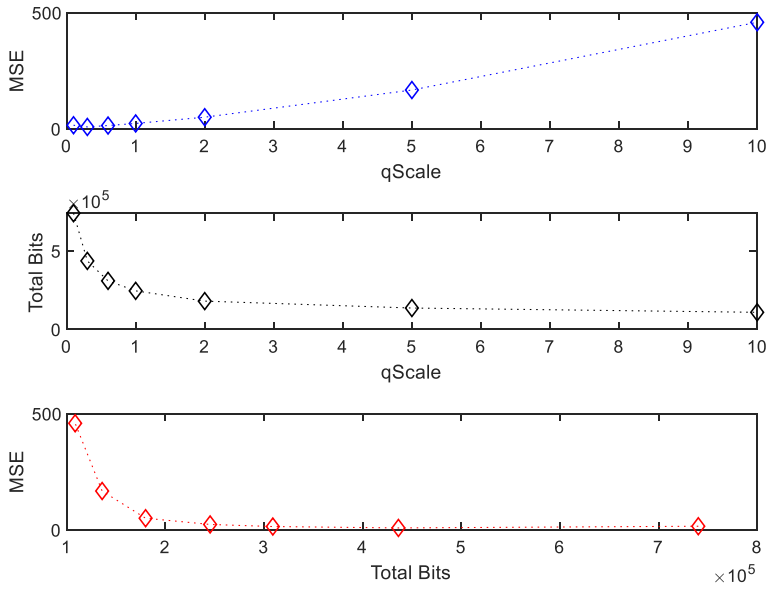
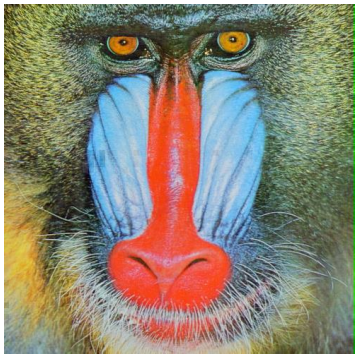
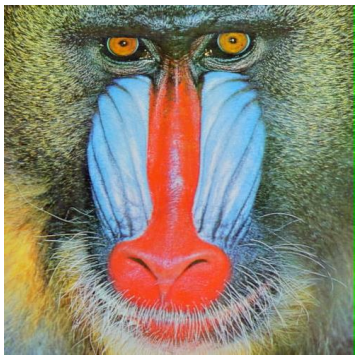
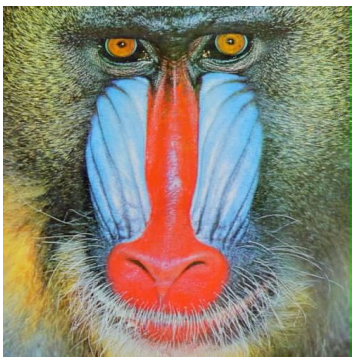
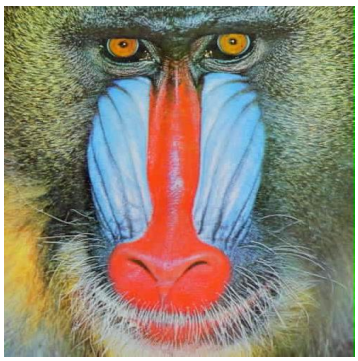


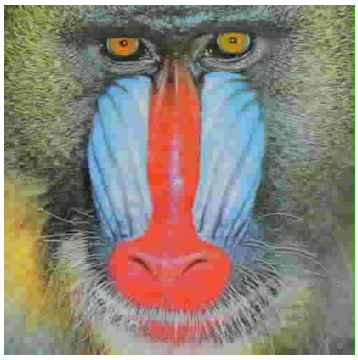
Image 2

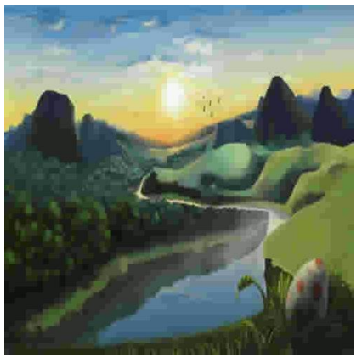


qScale	Image 1	Image 2
0.1		
0.3		
0.6		
1		

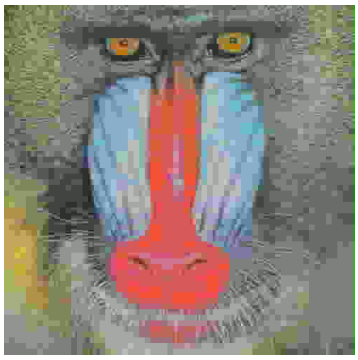
2

5





10



Όπως φαίνεται από τα διαγράμματα η αύξηση στο qScale αυξάνει το σφάλμα, πράγμα το οποίο γίνεται αντιληπτό και από τις δύο εικόνες καθώς η ποιότητα ελαττώνετε αρκετά, συγκεκριμένα για qScale > 1 αρχίζουμε να παρατηρούμε μεγάλη αλλοίωση των εικόνων.

Βέβαια από τα διαγράμματα φαίνεται επίσης ότι η αύξηση του qScale επιτρέπει την χρήση λιγότερων bits για την κωδικοποίηση. Για παράδειγμα στην πρώτη εικόνα για qScale = 0.1 απαιτούνται περίπου 1.5 εκατομμύρια bits, ενώ για qScale = 10 μόλις 80 χιλιάδες.

Η μέτρηση των bits έγινε αθροίζοντας τα μεγέθη των bitstreams κάθε κωδικοποιημένου block.

Υψηλές Συχνότητες

Πέραν της επιρροής του qScale ζητήθηκε να μελετηθεί και η επίδραση που έχουν οι υψηλές συχνότητες σε μια εικόνα και κατά πόσο αυτές γίνονται αντιληπτές από το ανθρώπινο μάτι.

Όπως έχει αναφερθεί οι συχνότητες που υπάρχουν σε ένα μπλοκ μιας φωτογραφίας εκφράζονται από τους συντελεστές DCT. Σε έναν πίνακα – μπλοκ DCT 8x8 κάθε ένα κελί αντιστοιχεί σε μια συχνότητα και ο αριθμός του κελιού εκφράζει το πόσο συντελεί η συγκεκριμένη συχνότητα στον σχηματισμό του συγκεκριμένο μπλοκ. Μια μετακίνηση σε δεξιότερη στήλη ή χαμηλότερη γραμμή οδηγεί σε κελί που αντιστοιχεί σε υψηλότερη συχνότητα, έτσι το κατώτερο και δεξιότερο κελί του πίνακα αντιστοιχεί στην μεγαλύτερη συχνότητα.

Για να μελετηθεί λοιπόν η επιρροή των υψηλών συχνοτήτων αρκεί να μηδενιστούν οι αντίστοιχοι συντελεστές DCT κατά την διαδικασία της κωδικοποίησης και στην συνέχεια να ανακατασκευαστεί η εικόνα με σκοπό να φανεί η επίδραση τους και αν θα μπορούσαν να αφαιρεθούν χωρίς όμως να υπάρξει αλλοίωση που μπορεί να γίνει αντιληπτή.

Από την εκφώνηση ζητείται να επαναληφτεί η διαδικασία πέντε φορές για κάθε εικόνα μηδενίζοντας κάθε φορά τους 20, 40, 50, 60 και 63 πλέον υψίσυχνους όρους. Επιλέγετε qScale = 1 και υποδειγματοληψία 4:2:0 και για της δύο εικόνες.

Για την παραπάνω διαδικασία δημιουργήθηκε το script **HighFrequenciesInfluense.m** που κωδικοποιεί και αποκωδικοποιεί τις εικόνες και έγιναν αλλαγές στην συνάρτηση **`quantizeJPEG`** ώστε να μηδενίζονται κάθε φορά οι κατάλληλοι συντελεστές.
Τα αποτελέσματα φαίνονται παρακάτω :

Μηδενικοί Υψίσυχνοι Όροι	**Image 1**	**Image 2**
20		
40		

50

60

63

Σημειώνεται ότι ο μηδενισμός των υψίσυχνων όρων μπορεί να γίνει με δύο τρόπους, είτε μηδενίζοντας κατευθείαν τους κατάλληλους όρους στους πίνακες κβαντισμένων DCT συντελεστών, είτε θέτοντας σε κατάλληλες θέσεις του πίνακα κβαντισμού μια τιμή, μεγαλύτερη ή ίση του μεγίστου του πίνακα DCT συντελεστών, ώστε μετά την διαίρεση και την στρογγυλοποίηση να μηδενίζονται οι αντίστοιχη όρη του πίνακα κβαντισμένων DCT συντελεστών. Οι υψίσυχνων όρη μου μηδενίζονται κάθε φορά είναι αυτοί που αντιστοιχούν στις 20, 40, 50, 60 και 63 τελευτές θέσεις ενός μονοδιάστατο πίνακα που προκύπτει από την zig – zag διαδικασία.

Όπως φαίνεται και από τις εικόνες η αφαίρεση μέχρι και των 40 υψίσυχνων όρων δεν γίνεται εύκολα αντιληπτή από το ανθρώπινο μάτι. Με την αφαίρεση των 50 υψίσυχνων όρων παρατηρείτε μια μικρή αλλοίωση κυρίως στην εικόνα 1, που παρουσιάζει μεγάλες συχνότητες (αλλαγές σε μικρές αποστάσεις) τόσο στην οριζόντια όσο και κατακόρυφη διεύθυνση. Με την αφαίρεση επιπλέον όρων οι λεπτομέρειες των εικόνων ελαττώνονται σε μεγάλο βαθμό καταστρέφοντας την ποιότητα.

Demo 2

Στο σημείο αυτό ζητείται να γίνει μετρήσεις της εντροπίας των εικόνων σε 3 σημεία :

- Spatial Domain
- Κβαντισμένη Συντελεστές DCT
- Μήκη διαδρομής

Γίνονται μετρήσεις και για τις δύο εικόνες και η κωδικοποίηση γίνεται με τα qScale και subimg του demo 1. Η ζητούμενη διαδικασία υλοποιείται στο αρχείο **demo2.m.**

Στο Spatial Domain η εντροπία μετράτε για κάθε συνιστώσα R G B ξεχωριστά. Στα άλλα δύο σημεία μετράτε η εντροπία των συνιστωσών Y Cb Cr ξεχωριστά. Σημειώνεται ότι η εντροπία των κβαντισμένων DCT συντελεστών και των πινάκων συμβόλων προκύπτει από την σύνθεση πινάκων που περιέχουν όλους τους συντελεστές DCT και όλα τα σύμβολα κάθε συνιστώσας Y Cb Cr και στην συνέχεια τον υπολογισμό εντροπίας σε αυτούς τους πίνακες.

Ο τρόπος υπολογισμού τις εντροπίας φαίνεται στο αρχείο.

Παρακάτω παρουσιάζονται οι τιμές εντροπίας για τα αναφερόμενα σημεία :

Image 1

Spatial Domain	7.6960 (R)	7.4730 (G)	7.7470 (B)
qDCT	1.7645 (Y)	0.5045 (Cb)	0.4578 (Cr)
RunSymbols	6.0829 (Y)	5.7053 (Cb)	5.6282 (Cr)

Image 2

Spatial Domain	7.6207 (R)	7.6725 (G)	7.5972 (B)
qDCT	0.2345 (Y)	0.1078 (Cb)	0.0767 (Cr)
RunSymbols	5.0359 (Y)	4.1767 (Cb)	3.4650 (Cr)

Όπως φαίνεται κατά την διαδικασία της κωδικοποίησης μιας εικόνας η εντροπία των συνιστωσών ελαττώνεται. Μεγαλύτερη μείωση εντροπίας παρουσιάζουν οι κβαντισμένοι συντελεστές DCT και παρατηρείται αύξηση στους πίνακες συμβόλων.

Η αύξηση της εντροπίας στους πίνακες συμβόλων οφείλεται στο γεγονός ότι αυξάνονται τα διαφορετικά σύμβολα επειδή κάθε σύμβολο αποτελείται από των αριθμό των μηδενικών και την τιμή του κβαντισμένου DCT συντελεστών.

Ωστόσο για να γίνει κατανοητή η συμπίεση που υπόκειται μια εικόνα πρέπει να ληφθεί υπόψιν και το πλήθος των συμβόλων προς κωδικοποίηση. Για παράδειγμα για την εικόνα 1 (subimg = 4 2 2 και qScale = 0.6) και ιδανική κωδικοποίηση το πλήθος των bits που απαιτείται, στο περίπου, σε κάθε στάδιο φαίνεται παρακάτω και κάνει αισθητή την συμπίεση που υφίσταται σε κάθε στάδιο:

Spatial Domain	$(7.7 + 7.5 + 7.7)*500*500 = 572.5 * 10^4$ bits
qDCT	$1.8*30752*8 + 0.5*15376*8 + 0.5*15376*8 = 56.6 * 10^4$ bits
RunSymbols	$6*60112 + 5.7*8286 + 5.6*7671 = 45.1 * 10^4$ bits