6/2/2020

Συστήματα Πολυμέσων και Εικονική Πραγματικότητα

Κωδικοποίηση – Αποκωδικοποίηση JPEG

ΓΕΩΡΓΙΟΣ ΜΠΑΛΑΟΥΡΑΣ (8861)

Περιεχόμενα

Παραδοχές Εργασίας

Για την υλοποίηση της εργασίας γίναν οι εξής παραδοχές.

- Οι πίνακες αναφοράς για τον Huffman θα οριστούν ως global μεταβλητές και θα αρχικοποιούνται από την συνάρτηση `ISO_Tables.m`.
- Οι πίνακες αναφοράς είναι σε μορφή $N \times 1$ διανύσματος, τύπου `String`.
- Για την σωστή κωδικοποίηση και αποκωδικοποίηση Huffman, έπρεπε να γνωρίζω τι είδους block (φωτεινότητας ή χρωματικότητας) δίνεται ως είσοδος. Έτσι, για να μην γίνει αλλαγή στα ορίσματα της συνάρτησης έγινε χρήση μιας ακόμα global μεταβλητής `isLuminance` (1 για φωτεινότητα και 0 για χρωματικότητα) που ελέγχει ποιοι πίνακες Huffman θα χρησιμοποιηθούν.
- Τα παραγόμενα `huffStream` μετά την εκτέλεση της `huffEnc` μετατρέπονται σε πίνακα από `uint8`, καθώς είναι η μόνη δομή του MATLAB για την αποθήκευση byte.
- Κατά την κλήση της `JPEGenc` στο πρώτο στοιχείο του cell array αποθηκεύονται τα τεχνικά χαρακτηριστικά της κωδικοποίησης. Στην αποθήκευση των πινάκων κβαντισμού, αποθηκεύονται για την ακρίβεια το $qScale * qTable$ και έτσι ο αποκβαντισμός γίνεται για τιμή $qScale = 1$.
- Για τον υπολογισμό των διαφορών μετρικών, χρησιμοποιούνται οι εξής τύποι:
  - **Mean Square Error**:
$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}\big(I(x,y) - I'(x,y)\big)^2$$
  - **Entropy**:
$$Entropy = \sum_i -f_i * \log_2(f_i)\ \ [per\ symbol]$$
  - **Compression Ratio**:
$$Compression\ Ratio = \frac{\overbrace{(M * N * 3)}^{\pi\lambda\acute{\eta}\theta o\varsigma\ bytes\ RGB\ \varepsilon\iota\kappa\acute{o}\nu\alpha\varsigma} * 8}{\sum \underbrace{length(huffStream)}_{\pi\lambda\acute{\eta}\theta o\varsigma\ bytes} * 8}$$

Παραδοτέα αρχεία Εργασίας

Εκτός των ζητούμενων συναρτήσεων και scripts, για καλύτερη οργάνωση των ζητούμενων και του κώδικα, δημιουργήθηκαν και τα εξής αρχεία.

▪ `changedTables.m`	Παράγει τους τροποποιημένους πίνακες κβαντισμού. Δέχεται ως όρισμα τον αριθμό των στοιχείων που θα τροποποιηθούν. Αν δωθεί ως όρισμα το 0, οι πίνακες είναι αυτοί που ορίζει το πρότυπο.
▪ `demoResults.m`	Παράγει τα ποιοτικά αποτελέσματα (Μηδενισμοί υψίσυχνων συντελεστών, MSE, Αριθμό bits, Compression Ratio) για κάθε εικόνα, για μια λίστα από $qScales$. Επιπλέον, παράγει τα κατάλληλα γραφήματα.
▪ `demoEntropy.m`	Για κάθε εικόνα, για κάθε δυνατό $subimg$, για μια λίστα από $qScales$ υπολογίζει την συνολική εντροπία για τα $Runlengths$. Τέλος, παράγει τα κατάλληλα γραφήματα.
▪ `ISO_Tables.m`	Αρχικοποιεί τους πίνακες αναφοράς για την κωδικοποίηση και την αποκωδικοποίηση Huffman.

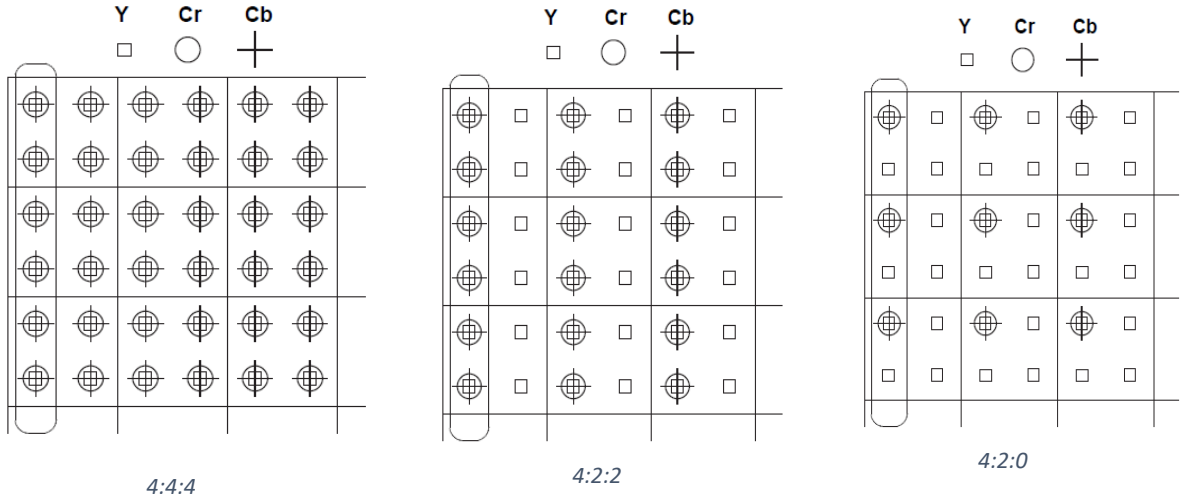
Αντικείμενο της εργασίας ήταν η υλοποίηση ενός κωδικοποιητή/αποκωδικοποιητή ακίνητης εικόνας, κατά το πρότυπο JPEG *(ISO/IEC 109181:1994)*. Συγκεκριμένα, θα υλοποιηθεί η εκδοχή baseline sequential DCT-based, με όσο το δυνατόν λιγότερες αποκλίσεις από το πρότυπο. Στο πρώτο μέρος, θα παρουσιαστεί η JPEG Library, η «βιβλιοθήκη» της εργασίας, όπου περιέχονται οι συναρτήσεις που υλοποιούν τα διάφορα τμήματα της εργασίας. Στο δεύτερο μέρος, έχει ως στόχο την ενσωμάτωση των συναρτήσεων του πρώτου μέρους σε δύο συναρτήσεις (κωδικοποίηση και αποκωδικοποίηση) και την εξαγωγή ποσοτικών και ποιοτικών συμπερασμάτων για την συμπίεση που επιτυγχάνεται.

Πρώτο μέρος

Μετατροπή RGB σε YCbCr και αντίστροφα

Σε πρώτο στάδιο και πριν αρχίσει η διαδικασία της κωδικοποίησης, έχουμε την προ-επεξεργασία της εικόνας, όπου μετατρέπεται από RGB σε YCbCr (*Luminance*, *Chrominance Blue*, *Chrominance Red*). Η μετατροπή αυτή επιτρέπει να χρησιμοποιηθεί διαφορετικός πίνακας κβαντισμού για την φωτεινότητα (*Luminance*) και για την χρωματικότητα (*Chrominance*). Επιπλέον, έτσι μπορεί να εφαρμοστεί η υποδειγματοληψία χρώματος, με σκοπό την μείωση του μεγέθους της κωδικοποίησης χωρίς να επηρεάζεται πολύ η πληροφορία που μεταφέρεται από την φωτογραφία[1]. Υποστηρίζονται τρεις περιπτώσεις υποδειγματοληψίας:

I. 4: 4: 4, όπου οι τρεις συνιστώσες δειγματοληπτούνται με τον ίδιο πίνακα πλέγματος.
II. 4: 2: 2, όπου οι συνιστώσες χρωματικότητας υποδειγματοληπτούνται οριζόντια και η ανάλυση τους μειώνεται κατά το ήμισυ σε σύγκριση με την φωτεινότητα.
III. 4: 2: 0, όπου οι συνιστώσες χρωματικότητας υποδειγματοληπτούνται με συντελεστή 2, τόσο οριζόντια όσο και κατακόρυφα. Έτσι η ανάλυση τους μειώνεται στο 1/4 σε σύγκριση με την φωτεινότητα.



Εικόνα 1: Χωροχρωνική δειγματοληψία

[1] Ο μηχανισμός της όρασης λειτουργεί ως βαθυπερατό χωρο-χρονικό φίλτρο και η ζώνη διέλευσης του είναι μικρότερη για τα σήματα που περιέχουν πληροφορίες χρωματικής αναλογίας (συνιστώσες Cb-Cr) απ' ότι για τα σήματα φωτεινότητας (Y).

Για την μετατροπή $RGB \leftrightarrow YCbCr$, χρησιμοποιήθηκαν οι εξής σχέσεις:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

και

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0 & 1.4020 \\ 1.0 & -0.3441 & -0.7141 \\ 1.0 & 1.7720 & 0.0 \end{bmatrix} \begin{bmatrix} Y \\ Cb - 128 \\ Cr - 128 \end{bmatrix}$$

Σημείωση: Πριν την εκτέλεση των επόμενων βημάτων, η εικόνα χωρίζεται σε blocks μεγέθους 8×8, για κάθε συνιστώσα. Αν το μέγεθος της εικόνας δεν είναι κατάλληλο, «κόβονται» οι περιττές γραμμές και στήλες, έως ότου το μέγεθος της εικόνας είναι πολλαπλάσιο του 16 (κατά γραμμές και στήλες).

Μετασχηματισμός DCT

Η εικόνα μετασχηματίζεται από μια χωρική αναπαράσταση (*spatial domain representation*) σε μια συχνοτική αναπαράσταση (*frequency domain representation*). Ουσιαστικά, τα περιεχόμενα της εικόνας μετατρέπονται σε μια μαθηματική αναπαράσταση, ένα άθροισμα ημιτονοειδών μοτίβων, με σκοπό την αποσυσχέτιση των δεδομένων και την επίτευξη μεγαλύτερης συμπίεσης. Για παράδειγμα, η δυαδική ακολουθία 101010 μπορεί να αναπαρασταθεί από ένα κύμα που επαναλαμβάνει κάθε δύο bit. Για τον μετασχηματισμό αυτό χρησιμοποιήθηκε ο DCT (*Discrete Cosine Transform*), καθώς έχει εξαιρετική ικανότητα συγκέντρωσης της πληροφορίας σε ένα μικρό πλήθος δειγμάτων.

Στα πλαίσια της εργασίας, για τον μετασχηματισμό χρησιμοποιήθηκαν οι συναρτήσεις dct2 και idct2, που παρέχονται από το MATLAB. Πριν την εκτέλεση της dct2 αφαιρείται το 128, καθώς η υλοποίηση του JPEG πρέπει να είναι εύκολα μεταφέρσιμες σε hardware[2]. Τέλος, στην αντίστροφη διαδικασία ξανά προστίθεται το 128.

Κβαντισμός και αποκβαντισμός

Μετά την εκτέλεση του DCT παράγεται ένα block (διάστασης 8×8), όπου οι συντελεστές που αντιστοιχούν σε χαμηλές συχνότητες «μαζεύονται» πάνω και αριστερά, στην αρχή του block. Όσο απομακρυνόμαστε, τοποθετούνται οι συντελεστές των υψηλών συχνοτήτων. Οι άνθρωποι είναι πιο ευαίσθητοι σε σφάλματα στην χαμηλόσυχνη πληροφορία, παρότι στην υψίσυχνη. Έτσι, προσπαθούμε να εκμεταλλευτούμε αυτήν την ιδιαιτερότητα, να θυσιάσουμε υψίσυχνες λεπτομέρειες για περαιτέρω συμπίεση χωρίς να επηρεαστεί η ποιότητα της εικόνας. Για να το επιτύχουμε, χρησιμοποιείται ο εξής τύπος σε κάθε block

$$Κβαντισμένοι\ Συντελεστές\ DCT = round\left(\frac{Συντελεστές\ DCT}{qScale * qTable}\right),$$

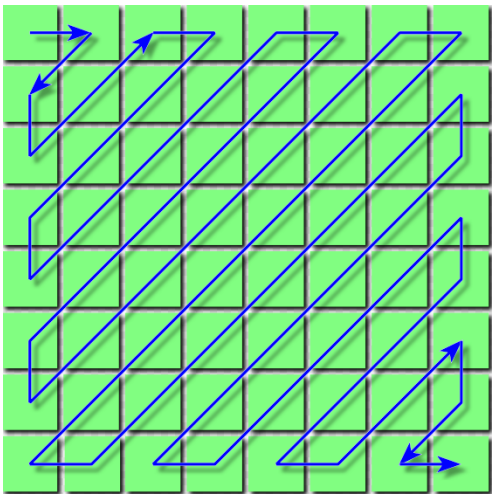
όπου ο κάθε συντελεστής DCT διαιρείται με την αντίστοιχη τιμή του πίνακα κβαντισμού ($qTable$), το $qScale$ ρυθμίζει την ποιότητα την κλίμακα κβαντισμού και έπειτα το αποτέλεσμα στρογγυλοποιείται στον κοντινότερο γείτονα.

[2] Εύρος επιτρεπτών τιμών -128 έως 127 και όχι 0 με 255.

Η σκέψη είναι πως οι μικροί συντελεστές όταν διαιρεθούν με μεγάλο διαιρέτη (λογικά) θα μηδενιστούν, ενώ οι σημαντικές τιμές θα μείνουν ανεπηρέαστες. Επιπλέον, είναι προφανές πως για την αποκβαντοποίηση των τιμών ακολουθείται η αντίστροφη διαδικασία. Τέλος, αξίζει να σημειωθεί πως για τους συντελεστές φωτεινότητας και χρωματικότητας χρησιμοποιούνται διαφορετικοί πίνακες κβαντισμού[3].

Μήκη Διαδρομής (Runlength)

Επόμενο στάδιο του προτύπου είναι η εύρεση των μηκών διαδρομής. Η διαδικασία του κβαντισμού – ευελπιστούμε- να έχει παράγει πολλά μηδενικά, όπου όσο μικρότερη η ζητούμενη ανάλυση τόσο περισσότερα μηδενικά θα παραχθούν. Αν αναδιατάξουμε τα στοιχεία, ξεκινώντας από την πάνω αριστερή γωνία, σε ένα διάνυσμα 64 στοιχείων σε zig-zag μοτίβο (εικόνα) θα στοιχίσουμε τα στοιχεία από τους χαμηλόσυχνους συντελεστές στους πιο υψίσυχνους.



Εικόνα 2: Zig-Zag μοτίβο

Στο διάνυσμα που παράχθηκε η πρώτη τιμή αποτελεί την DC συνιστώσα, ενώ οι υπόλοιπες συνθέτουν τις AC συνιστώσες. Για την κωδικοποίηση του DC όρου θα χρησιμοποιηθεί διαφορετική διαδικασία απ' ότι για τους AC. Στην περίπτωση του DC όρου, στην πραγματικότητα θα κωδικοποιηθεί η διαφορά (*DPCM - Differential Pulse Code Modulation*)

$$DIFF = DC_i - PRED$$

όπου DC_i ο (κβαντισμένος) DC όρος του τωρινού block και $PRED$, ο αντίστοιχος όρος του προηγούμενου προς κωδικοποίηση block (ίδιας συνιστώσας).

Για τις AC συνιστώσες μετράμε πόσα μηδενικά προηγήθηκαν των τιμών τους και δημιουργούμε τα ζεύγη

$$< \text{precedingZeros}, \text{quantSymbol} >$$

με σκοπό να δημιουργήσουμε έναν πίνακα μεγέθους $R \times 2$.[4]

[3] Ορίζονται πλήρως από το πρότυπο.

[4] Στην πρώτη γραμμή του πίνακα θα τοποθετηθεί η κωδικοποίηση του DC όρου ως $< 0 , DIFF >$.

Σε αυτό το σημείο να σημειωθεί, πως στην συνέχεια βρίσκεται το στάδιο της Huffman κωδικοποίησης και μέσω αυτής ορίζονται δύο ειδικές περιπτώσεις. Πιο συγκεκριμένα

1. Δεν μπορούν να υπάρξουν περισσότερα από 16 συνεχόμενα μηδενικά (*ZRL*). Αν βρεθεί τέτοια περίπτωση πρέπει τα precedingZeros να διαχωριστούν. Για παράδειγμα αν βρέθηκαν 22 μηδενικά πριν την τιμή 1, τότε ο πίνακας των μηκών διαδρομής θα έχει την μορφή
$$\begin{bmatrix} \dots & \dots \\ 15 & 0 \\ 6 & 1 \\ \dots & \dots \end{bmatrix}$$
2. Αν τα συνεχόμενα μηδενικά βρίσκονται έως το τέλος του κβαντισμένου block, τότε τοποθετείται το ζεύγος του EOB (*EndOfBlock*) στον πίνακα των μηκών διαδρομής, δηλαδή $< 0 , 0 >$.

Η παραπάνω διαδικασία έχει ως σκοπό να συγκεντρώσει τα μηδενικά σε ομάδες και να μειώσει περαιτέρω τα προς κωδικοποίηση απαιτούμενα κομμάτια πληροφορίας.

Κωδικοποίηση και αποκωδικοποίηση Huffman

Τελευταίο στάδιο του προτύπου είναι η κωδικοποίηση Huffman. Πρόκειται για κώδικα μεταβλητού μήκους, όπου αναθέτει στα σύμβολα κωδικές λέξεις μήκους αντιστρόφως αυξανόμενου με την πιθανότητα εμφάνισης τους. Στα πλαίσια της εργασίας δεν υλοποιήθηκε η κωδικοποίηση Huffman από την αρχή, αλλά χρησιμοποιήθηκαν οι πίνακες που ορίζει το πρότυπο, ως πίνακες αναφοράς. Αξίζει να σημειωθεί πως οι πίνακες αντιστοίχισης κωδικών λέξεων και συμβόλων είναι διαφορετικοί τόσο για τους DC και AC όρους όσο και για την χρωματικότητα και την φωτεινότητα.

Υλοποίηση στο MATLAB

Η κωδικοποίηση Huffman δέχεται ως είσοδο τον πίνακα με τα μήκη διαδρομής, ωστόσο δεν κωδικοποιεί ακριβώς τα περιεχόμενα του. Πιο συγκεκριμένα η διαδικασία που ακολουθείται είναι η εξής:

- Για τον DC όρο πρέπει να βρεθεί η Κατηγορία στην οποία ανήκει η $DIFF$. Για την Κατηγορία ισχύει
$$-2^{Category} < Symbol < 2^{Category}$$
δηλαδή για τον υπολογισμό του $Category$ μπορεί να γίνει χρήση της εξής κλαδικής σχέσης
$$Category = \begin{cases} 0, & Symbol = 0 \\ \lfloor \log_2(|Symbol|) \rfloor + 1, & elsewhere \end{cases}$$
- Έπειτα, πρέπει να υπάρχει δυνατότητα να διαχωρίζονται διαφορετικά $Symbols$ τα οποία κατατάσσονται στην ίδια Κατηγορία. Αυτό συμβαίνει με τα επιπλέον bits που τοποθετούνται μετά την κωδική λέξης της Κατηγορίας. Για να βρεθούν αυτά τα επιπλέον bits, σκεφτόμαστε ως εξής:
    - $Category = 0$, δεν υπάρχουν επιπλέον bits.
    - $Category \neq 0 , Symbol > 0$, τα επιπλέον bits είναι η δυαδική αναπαράσταση του $Symbol$.
    - $Category \neq 0 , Symbol < 0$, τα επιπλέον bits είναι το συμπλήρωμα ως προς 1 της δυαδικής αναπαράστασης του $Symbol$.
- Για τους AC όρους ακολουθείται παρόμοια διαδικασία, με τις εξής διαφοροποιήσεις:
    - Δεν υπάρχει $Category = 0$.
    - Για τα μήκη διαδρομής $[0 , 0]$ (EOB) και $[15, 0]$ (ZRL) δεν υπάρχουν επιπλέον bits.

- Τέλος, για την εύρεση του huffStream κάθε block πρέπει να γίνει αντιστοίχιση των ζευγών $<\text{precedingZeros}, Category>$ που βρέθηκαν στους κατάλληλους πίνακες αναφοράς, για την εύρεση των κωδικών λέξεων που τους αντιστοιχούν. Πιο συγκεκριμένα,
    - Για τους DC όρους η θέση, $index$, της κωδικής λέξης είναι $index = Category + 1$.
    - Για τους AC όρους ισχύει $index = \text{precedingZeros} * 10 + Category + offset$, με
$$offset = \begin{cases} 1, & \text{precedingZeros} \neq 15 \\ 2, & \text{precedingZeros} = 15 \end{cases}$$

Για την αποκωδικοποίηση του huffStream, εκμεταλλευόμαστε την ιδιότητα του κώδικα Huffman να είναι απροθηματικός, δηλαδή καμία κωδική λέξη δεν αποτελεί πρόθημα άλλης κωδικής λέξης. Πιο συγκεκριμένα:

- Εκτελείται μια αναζήτηση πάνω στο huffStream για την εύρεση της κωδικής λέξης του DC όρου. Όταν βρεθεί ακολουθείται η αντίστροφη διαδικασία που περιεγράφηκε παραπάνω για την εύρεση της Κατηγορίας και των επιπλέον bits.
- Από το σημείο που τελείωσαν τα ψηφία του κώδικα που αφορούν τον DC όρο, αρχίζει η αναζήτηση για όλους τους AC όρους και των επιπλέον bits τους. Για να «γυρίσουμε» από την θέση του πίνακα που βρέθηκε η κωδική λέξη, έστω $index$, ακολουθείται η εξής διαδικασία:
    - $index = index - offset$, με $offset = \begin{cases} 1, & index < 152 \\ 2, & index > 152 \end{cases}$
    - Υπολογίζεται το $Remainings = index \ mod \ 10$, όπου

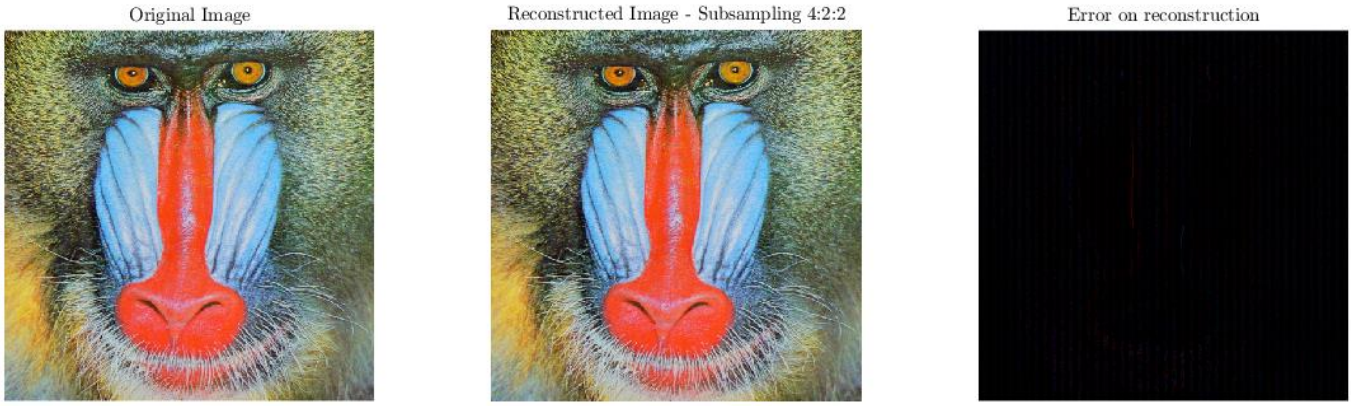
$$Remainings \neq 0, \quad Category = Remainings, \quad \text{precedingZeros} = \left\lfloor \frac{index}{10} \right\rfloor$$

$$Remainings = 0, \quad Category = 10, \quad \text{precedingZeros} = \left\lfloor \frac{index}{10} \right\rfloor - 1$$

- Η διαδικασία ολοκληρώνεται όταν βρεθεί το EOB ή διαβαστεί ολόκληρο το huffStream.

Demo 1

Σε πρώτο στάδιο, ελέγχεται η μετατροπή από RGB σε YCbCr και πάλι πίσω. Για την πρώτη εικόνα, χρησιμοποιήθηκε υποδειγματοληψία 4: 2: 2

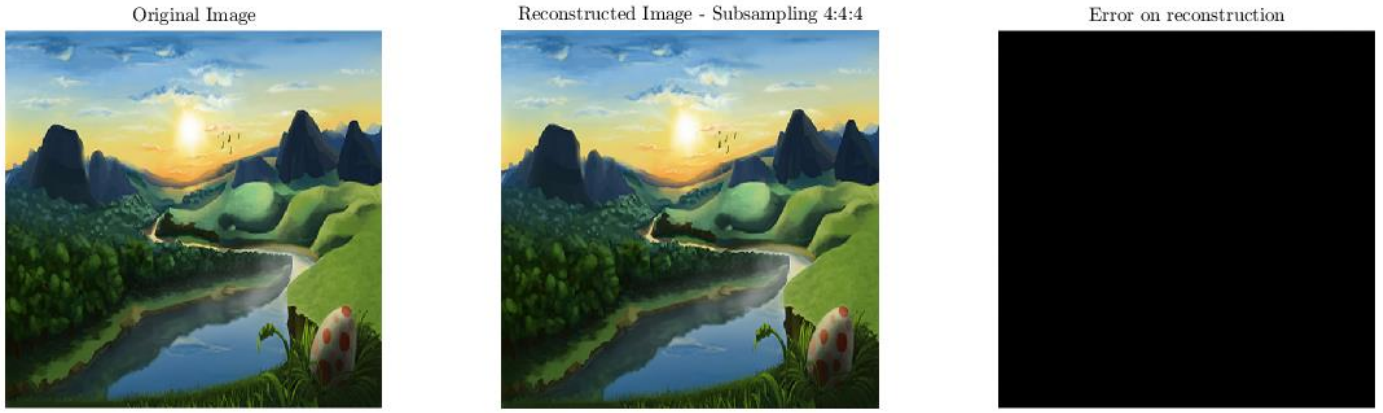


Εικόνα 3: Πρώτη εικόνα - Υποδειγματοληψία 4: 2: 2 και σφάλμα ανακατασκευής [demo 1]

όπου παρατηρούμε πως η εικόνα ανακατασκευάστηκε σωστά, με το σφάλμα, που παρουσιάζεται στην δεξιά εικόνα, να οφείλεται στη παρεμβολή κατά την επιστροφή στον RGB χώρο για τον υπολογισμό των

τιμών που έλειπαν. Χρησιμοποιήθηκε η παρεμβολή του κοντινότερου γείτονα, με τις τιμές που έλειπαν –λόγω της υποδειγματοληψίας- να υπολογίζονται ως η τιμή που είχε το pixel στα αριστερά τους. Στην περίπτωση της υποδειγματοληψίας $4{:}\,2{:}\,0$, που δεν παρουσιάζεται εδώ, η παρεμβολή γίνεται σε παράθυρο μεγέθους 2×2, με τις τιμές να ορίζονται ως η τιμή του πάνω αριστερά pixel.

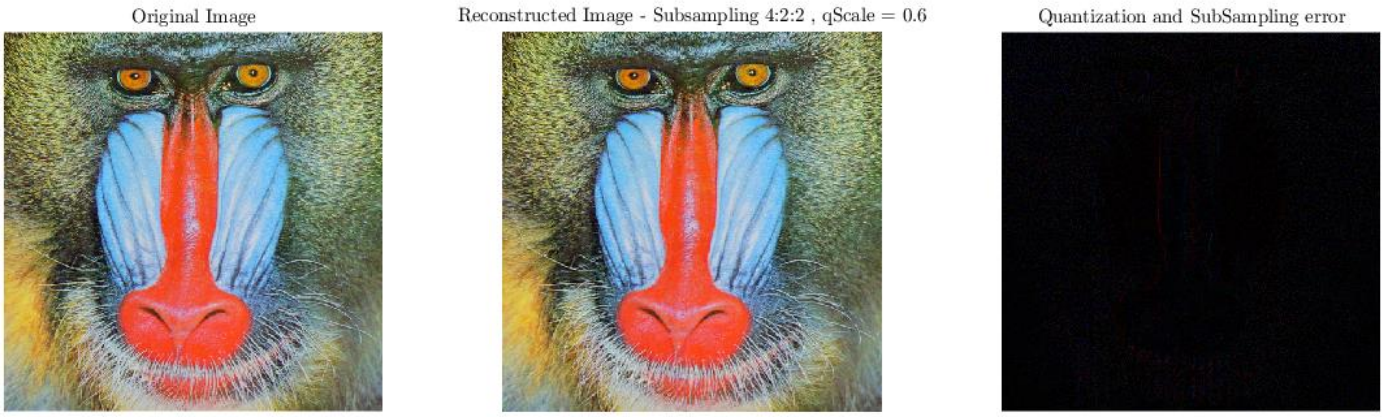
Για την δεύτερη εικόνα, χρησιμοποιήθηκε υποδειγματοληψία $4{:}\,4{:}\,4$



Εικόνα 4: Δεύτερη εικόνα - Υποδειγματοληψία $4{:}\,4{:}\,4$ και σφάλμα ανακατασκευής [demo 1]

όπου παρατηρούμε πως η εικόνα ανακατασκευάστηκε τέλεια, με το σφάλμα, που παρουσιάζεται στην δεξιά εικόνα, να είναι μηδενικό αφού δεν υπήρξε παρεμβολή για τον υπολογισμό τιμών.

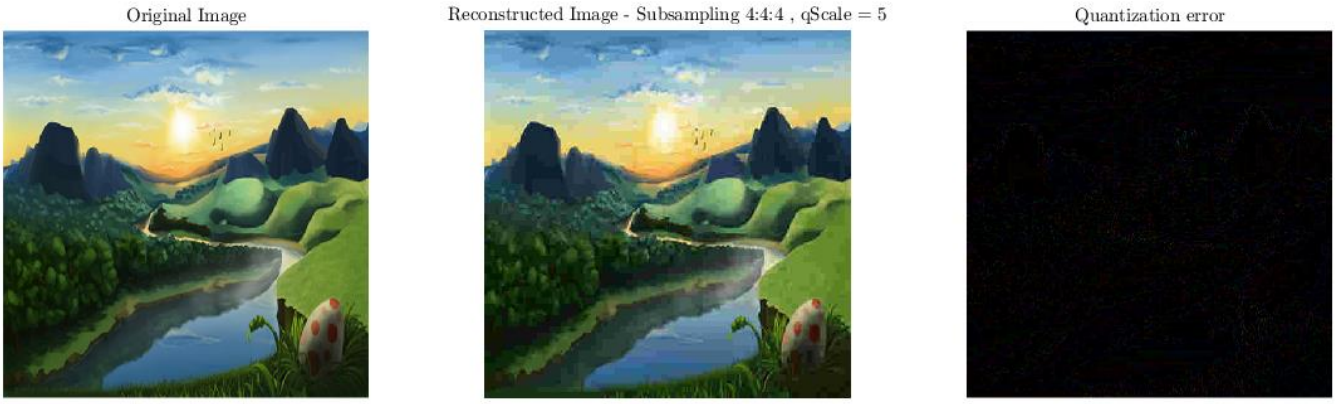
Στην συνέχεια, για τις ίδιες εικόνες και τύπο υποδειγματοληψίας, θα υπολογιστούν οι εικόνες αφού εκτελεστεί και το βήμα του κβαντισμού. Για την πρώτη εικόνα, χρησιμοποιήθηκε $qScale = 0.6$



Εικόνα 5: Πρώτη εικόνα - Υποδειγματοληψία $4{:}\,2{:}\,2$, $qScale = 0.6$ και σφάλμα ανακατασκευής [demo 1]

Σε αυτήν την περίπτωση το σφάλμα είναι μεγαλύτερο, καθώς εισάγεται και το σφάλμα κβαντισμού. Το μέσο τετραγωνικό σφάλμα είναι ίσο με $MSE = 43.4974$ και το οπτικό αποτέλεσμα είναι σωστό.

Για την δεύτερη εικόνα, χρησιμοποιήθηκε $qScale = 5$



Εικόνα 6: Δεύτερη εικόνα - Υποδειγματοληψία $4:4:4$*,* $qScale = 5$ *και σφάλμα ανακατασκευής [demo 1]*

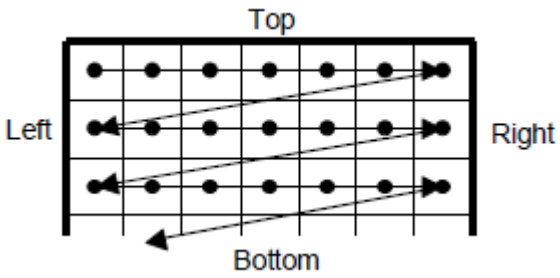
Σε αυτήν την περίπτωση το σφάλμα δεν είναι μηδενικό, καθώς εισάγεται και το σφάλμα κβαντισμού. Το μέσο τετραγωνικό σφάλμα είναι ίσο με $MSE = 28.2781$ και το οπτικό αποτέλεσμα είναι σωστό.

Δεύτερο Μέρος

Το δεύτερο μέρος περιλαμβάνει την ενοποίηση των προηγούμενων συναρτήσεων, για την πλήρη κωδικοποίηση και αποκωδικοποίηση εικόνων. Η διαδικασία που ακολουθείται είναι η εξής:

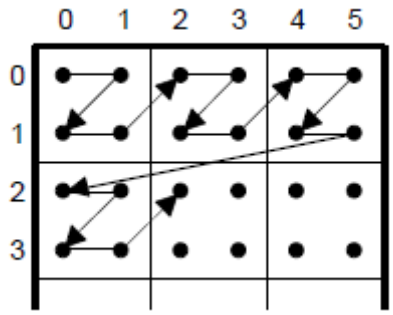
1. Μετατρέπεται το σύνολο της εικόνας από RGB σε YCbCr.
2. Χωρίζεται η εικόνα σε blocks διάστασης 8×8, για κάθε συνιστώσα χωριστά (Y, Cb, Cr).
3. Κάθε block κάθε συνιστώσας, για την κωδικοποίηση ακολουθεί την εξής πορεία.
    a. Μετασχηματισμός DCT.
    b. Κβαντισμός συντελεστών DCT.
    c. Υπολογισμός μηκών διαδρομής.
    d. Κωδικοποίηση Huffman.
4. Αποθήκευση όλων των N blocks, σε ένα cell array (μεγέθους $N \times 1$). Στο πρώτο κελί του ίδιου cell array, αποθηκεύονται τα τεχνικά χαρακτηριστικά της κωδικοποίησης (πίνακες κβαντισμού και πίνακες αναφοράς για τον κώδικα Huffman).
5. Για την αποκωδικοποίηση ακολουθείται η αντίστροφη πορεία.
    a. Αποκωδικοποίηση Huffman.
    b. Από τα μήκη διαδρομής, υπολογισμός των κβαντισμένων συντελεστών DCT.
    c. Αποκβαντισμός των συντελεστών DCT.
    d. Εφαρμογή αντίστροφου μετασχηματισμού DCT.
6. Μετατροπή του συνόλου της εικόνας από YCbCr σε RGB.

Σε αυτό το σημείο πρέπει να διευκρινιστεί πως η σειρά κωδικοποίησης των blocks δεν είναι πάντα η ίδια. Στην πληθώρα των περιπτώσεων και για όλες τι συνιστώσες, η σειρά κωδικοποίησης των blocks είναι κατά σειρά ξεκινώντας από το πρώτο block, πάνω και αριστερά.



Εικόνα 7: Συνήθης σειρά κωδικοποίησης block

Η σειρά αλλάζει μόνο για την συνιστώσα Y στην περίπτωση υποδειγματοληψίας $4:2:0$. Πρέπει τα blocks να κωδικοποιηθούν σε Z μορφή (παρουσιάζεται παρακάτω)



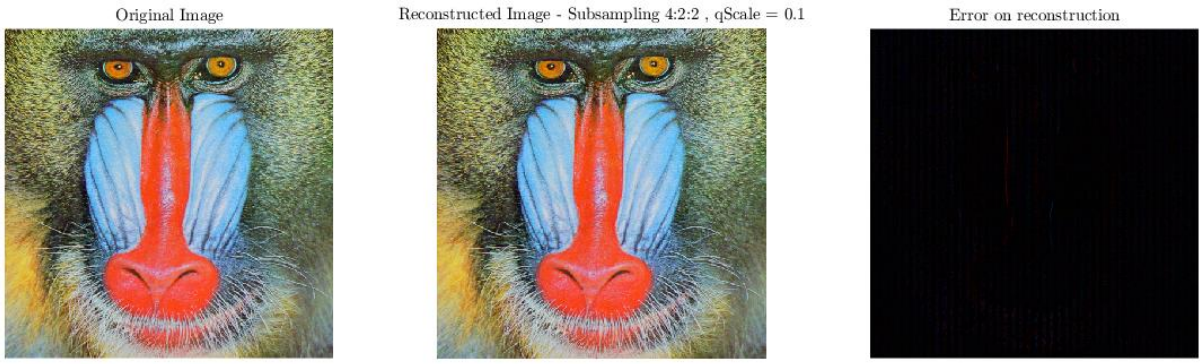
Εικόνα 8: Σειρά κωδικοποίησης block για υποδειγματοληψία $4:2:0$ στην συνιστώσα Y

Ποιοτικά αποτελέσματα

Διάφορες τιμές qScale – Πρώτη εικόνα

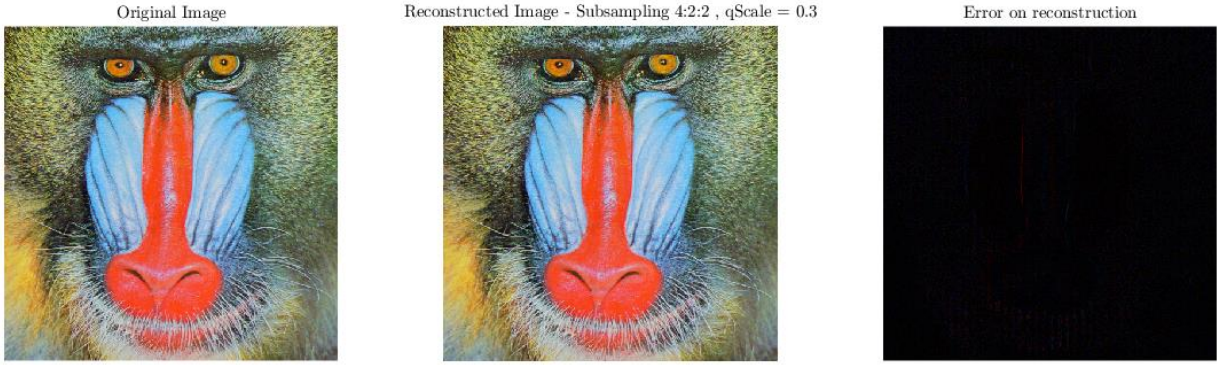
Εκτελώντας την παραπάνω σειρά βημάτων, παρουσιάζονται τα αποτελέσματα για διαφορές τιμές του qScale στην ποιότητα της πρώτης εικόνας. Δεν διευκρινίστηκε η υποδειγματοληψία, έτσι χρησιμοποιήθηκε $4:2:2$, από το demo1.

- $qScale = 0.1$



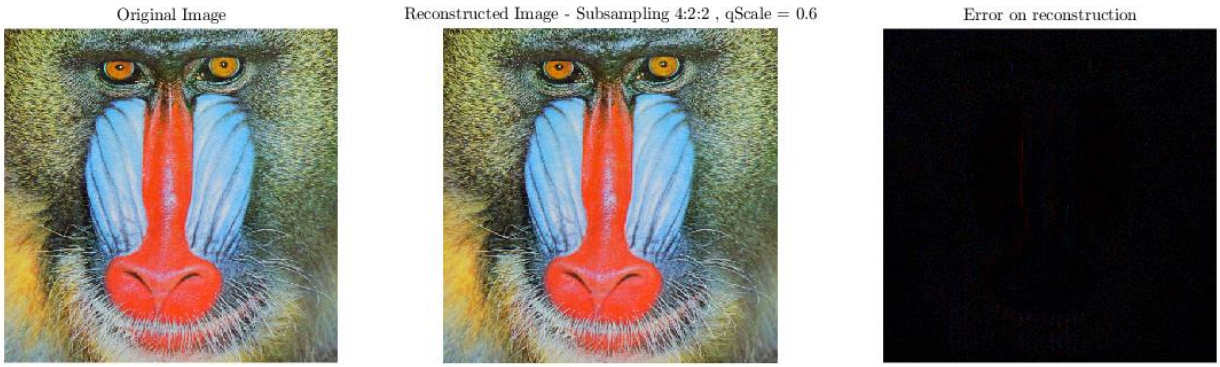
Εικόνα 9: Πρώτη εικόνα - Υποδειγματοληψία $4:2:2$, $qScale = 0.1$ και σφάλμα ανακατασκευής [demo Results]

- $\underline{qScale = 0.3}$



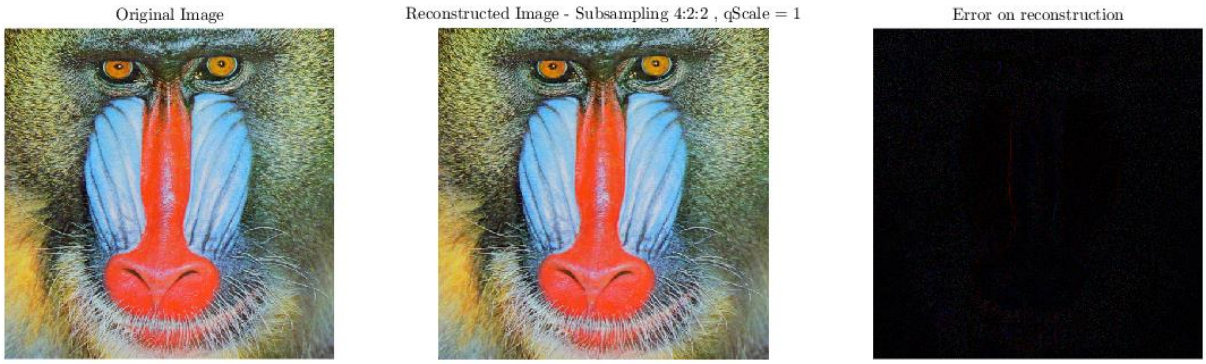
Εικόνα 10: Πρώτη εικόνα - Υποδειγματοληψία $4{:}\,2{:}\,2$*,* $qScale = 0.3$ *και σφάλμα ανακατασκευής [demo Results]*

- $\underline{qScale = 0.6}$



Εικόνα 11: Πρώτη εικόνα - Υποδειγματοληψία $4{:}\,2{:}\,2$*,* $qScale = 0.6$ *και σφάλμα ανακατασκευής [demo Results]*

- $\underline{qScale = 1}$

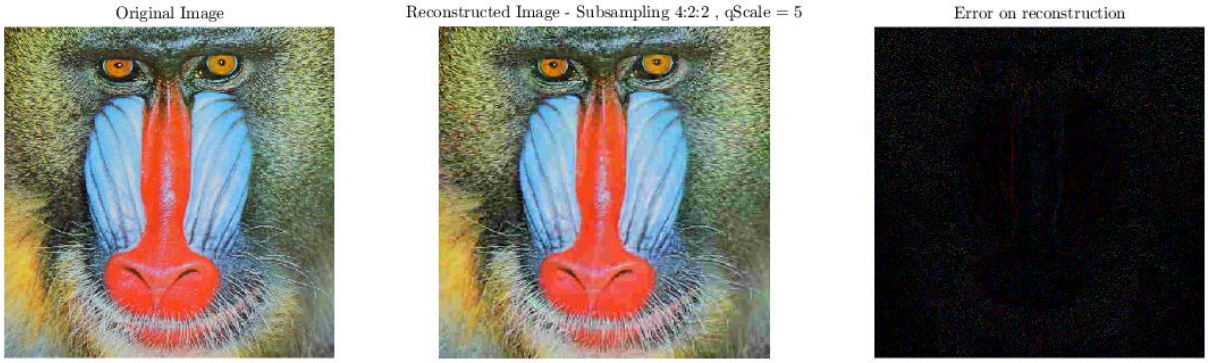


Εικόνα 12: Πρώτη εικόνα - Υποδειγματοληψία $4{:}\,2{:}\,2$*,* $qScale = 1$ *και σφάλμα ανακατασκευής [demo Results]*

- $\underline{qScale = 2}$

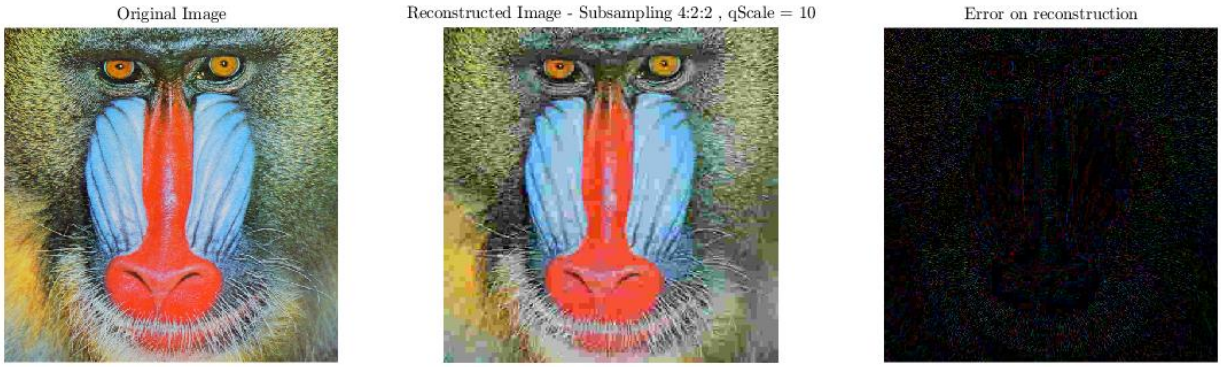
Εικόνα 13: Πρώτη εικόνα - Υποδειγματοληψία $4{:}\,2{:}\,2$, $qScale = 2$ *και σφάλμα ανακατασκευής [demo Results]*

- $\underline{qScale = 5}$



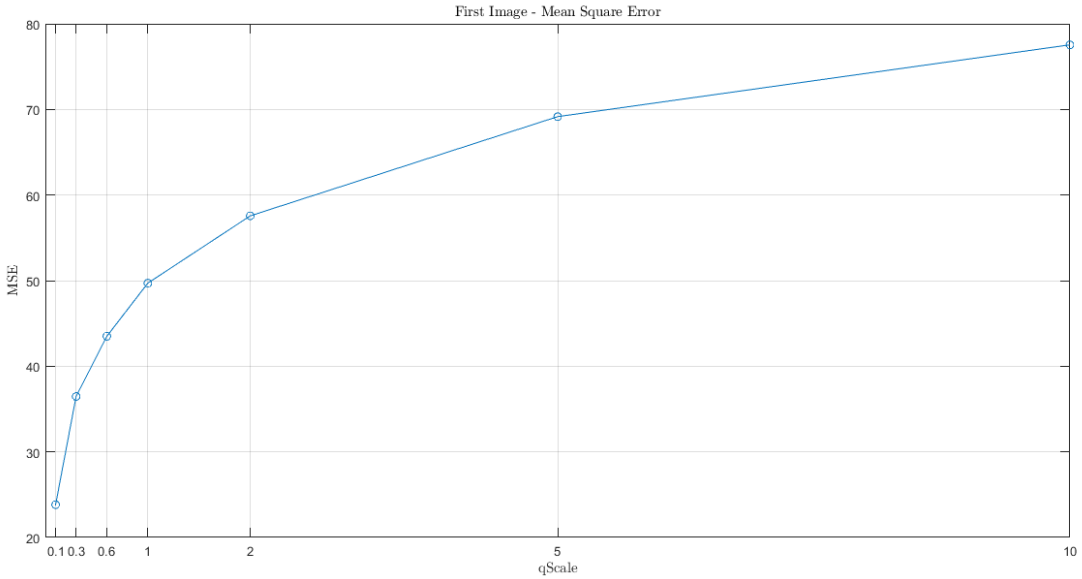
Εικόνα 14: Πρώτη εικόνα - Υποδειγματοληψία $4{:}\,2{:}\,2$, $qScale = 5$ *και σφάλμα ανακατασκευής [demo Results]*

- $\underline{qScale = 10}$



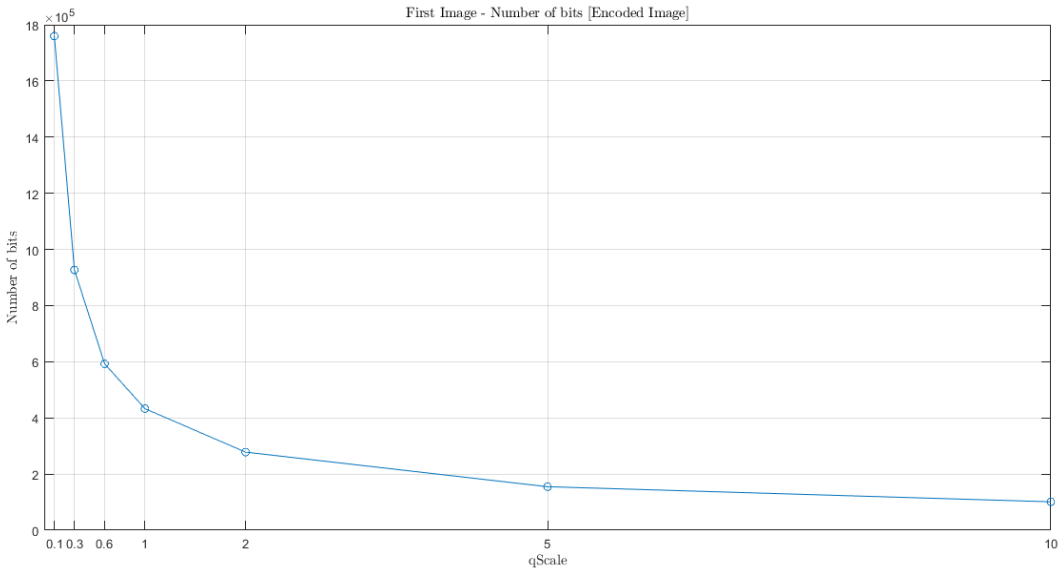
Εικόνα 15: Πρώτη εικόνα - Υποδειγματοληψία $4{:}\,2{:}\,2$, $qScale = 10$ *και σφάλμα ανακατασκευής [demo Results]*

Παρατηρούμε πως όσο αυξάνει το $qScale$, τόσο χαλάει το οπτικό αποτέλεσμα, ειδικά στις λεπτομέρειες και στις γρήγορες μεταβολές της εικόνας (πχ. στο κίτρινο του τριχώματος). Η παρατήρηση μας επιβεβαιώνεται και από την αύξηση του MSE όσο αυξάνεται το $qScale$.



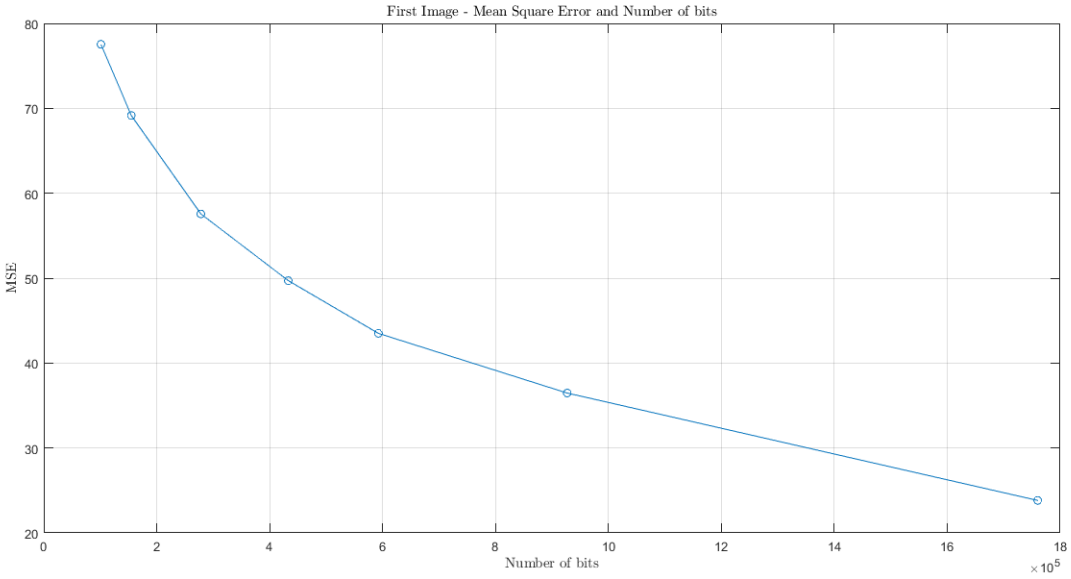
Εικόνα 16: MSE και qScale [Πρώτη Εικόνα]

Ωστόσο, πρέπει να σημειωθεί πως με την αύξηση του $qScale$, παρατηρείται μείωση του αριθμού των bits της κωδικοποιημένης εικόνας.



Εικόνα 17: Αριθμός bits και qScale [Πρώτη Εικόνα]

Τέλος, δίνεται το γράφημα που δείχνει την σχέση μεταξύ του MSE και του αριθμού των bits της κωδικοποιημένης εικόνας.



Εικόνα 18: Αριθμός bits και MSE [Πρώτη Εικόνα]

Η διαίσθηση μας, πως η αύξηση του αριθμού των bits θα οδηγήσει σε μείωση του MSE επιβεβαιώνεται.

Διάφορες τιμές qScale – Δεύτερη εικόνα

Στην συνέχεια, η ίδια διαδικασία επαναλαμβάνεται για την δεύτερη εικόνα. Δεν διευκρινίστηκε η υποδειγματοληψία, έτσι χρησιμοποιήθηκε 4: 4: 4, από το demo1.

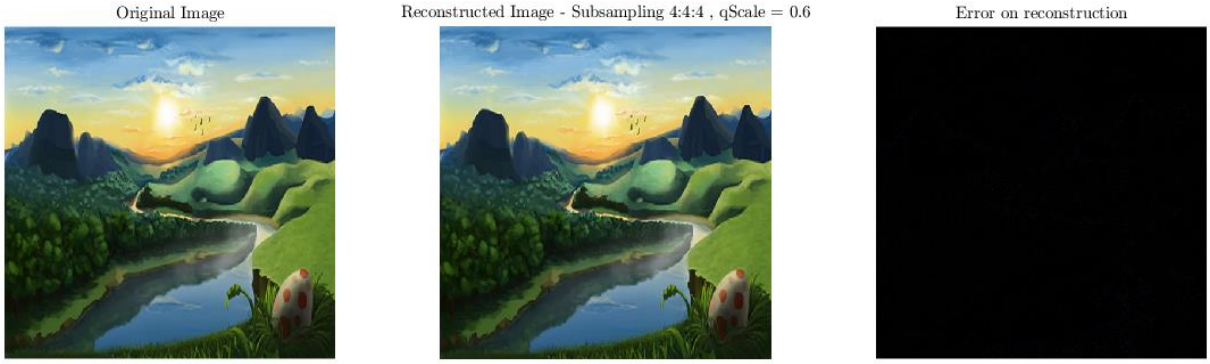
- $\underline{qScale = 0.1}$

Εικόνα 19: Δεύτερη εικόνα - Υποδειγματοληψία 4: 4: 4, $qScale = 0.1$ και σφάλμα ανακατασκευής [demo Results]

- $\underline{qScale = 0.3}$

Εικόνα 20: Δεύτερη εικόνα - Υποδειγματοληψία 4: 4: 4, $qScale = 0.3$ και σφάλμα ανακατασκευής [demo Results]

- $\underline{qScale = 0.6}$

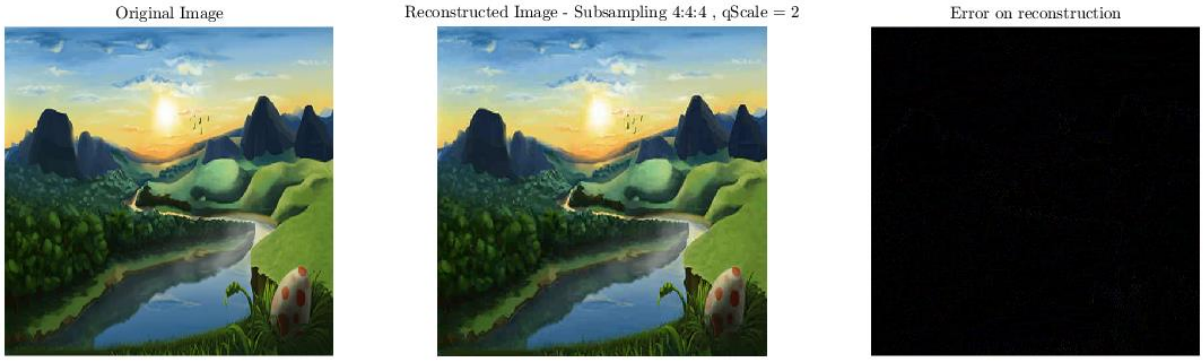


Εικόνα 21: Δεύτερη εικόνα - Υποδειγματοληψία 4: 4: 4, $qScale = 0.6$ και σφάλμα ανακατασκευής [demo Results]

- $\underline{qScale = 1}$

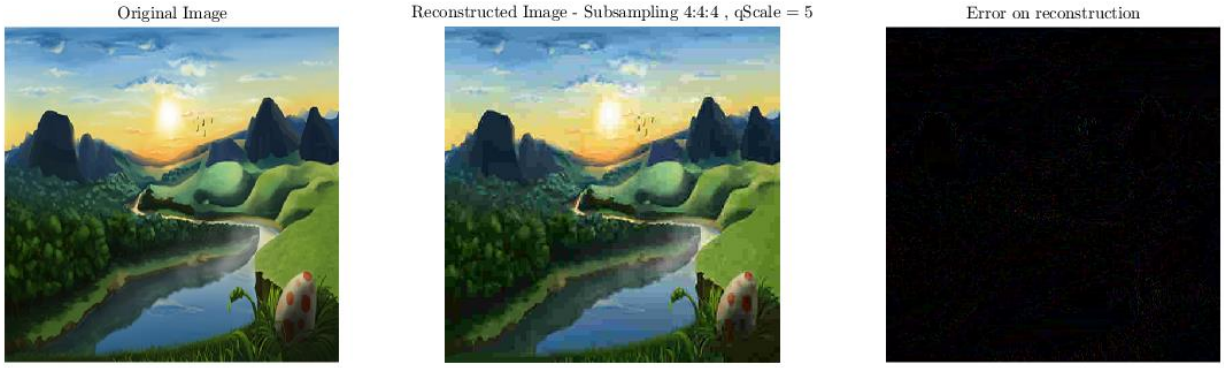
Εικόνα 22: Δεύτερη εικόνα - Υποδειγματοληψία 4: 4: 4, $qScale = 1$ και σφάλμα ανακατασκευής [demo Results]

- $\underline{qScale = 2}$



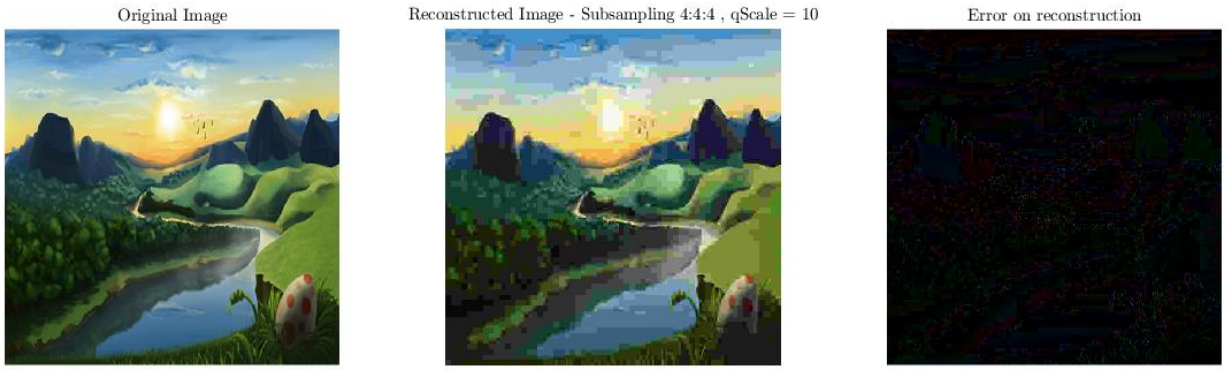
Εικόνα 23: Δεύτερη εικόνα - Υποδειγματοληψία $4{:}\,4{:}\,4$*,* $qScale = 2$ *και σφάλμα ανακατασκευής [demo Results]*

- $\underline{qScale = 5}$



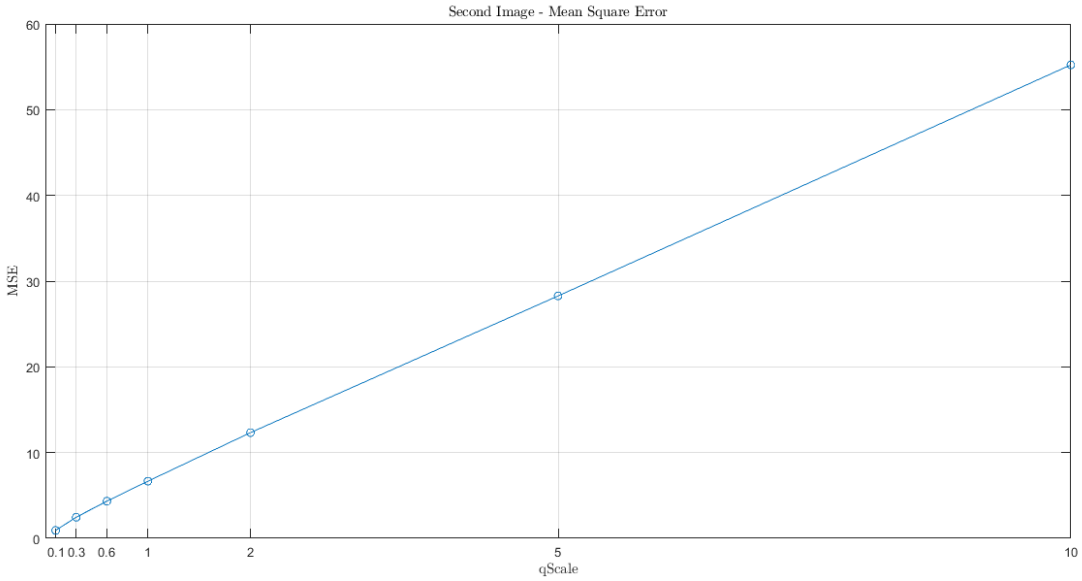
Εικόνα 24: Δεύτερη εικόνα - Υποδειγματοληψία $4{:}\,4{:}\,4$*,* $qScale = 5$ *και σφάλμα ανακατασκευής [demo Results]*

- $\underline{qScale = 10}$



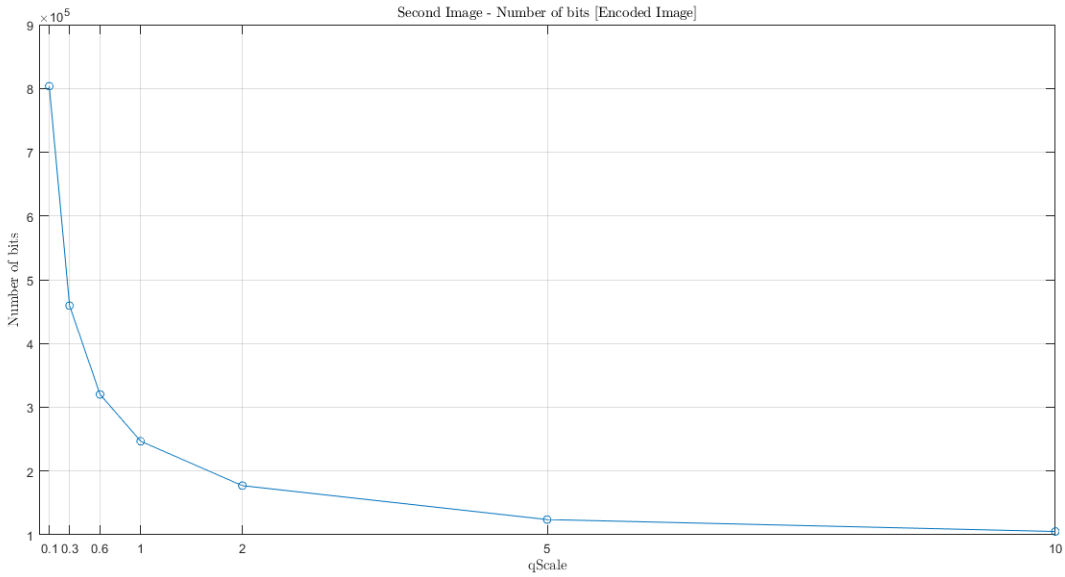
Εικόνα 25: Δεύτερη εικόνα - Υποδειγματοληψία $4{:}\,4{:}\,4$*,* $qScale = 10$ *και σφάλμα ανακατασκευής [demo Results]*

Παρατηρούμε –ξανά- πως όσο αυξάνει το $qScale$, τόσο χαλάει το οπτικό αποτέλεσμα, ειδικά στις λεπτομέρειες και στις λεπτομέρειες της εικόνας (πχ. στην αίσθηση του βάθους που έδινε η φωτογραφία και στην ποιότητα των χρωμάτων). Η παρατήρηση μας επιβεβαιώνεται και από την αύξηση του MSE όσο αυξάνεται το $qScale$.



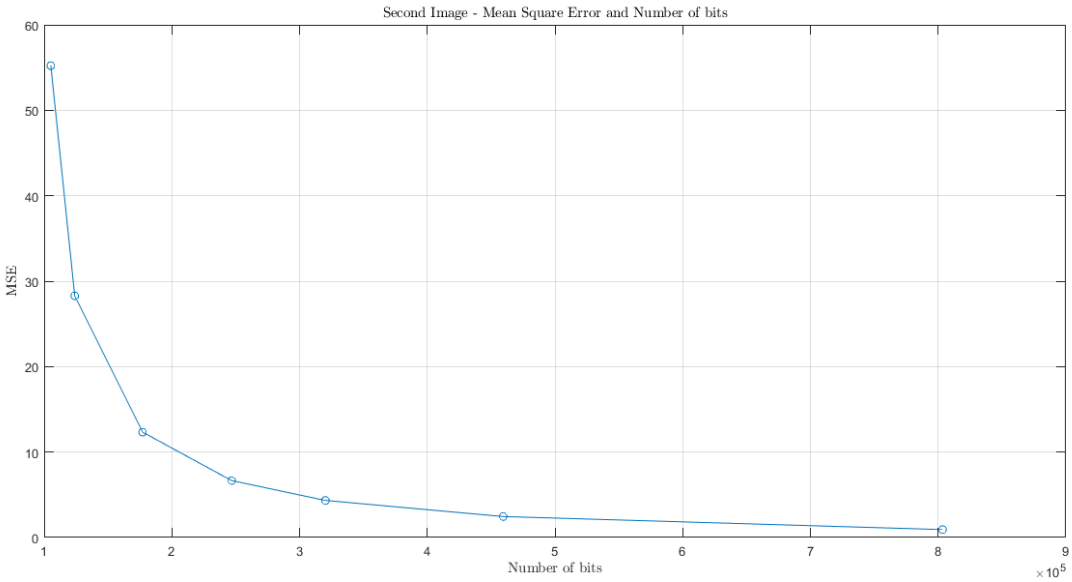
Εικόνα 26: MSE και qScale [Δεύτερη Εικόνα]

Ωστόσο, πρέπει να σημειωθεί πως με την αύξηση του $qScale$, παρατηρείται μείωση του αριθμού των bits της κωδικοποιημένης εικόνας.



Εικόνα 27: Αριθμός bits και qScale [Δεύτερη Εικόνα]

Τέλος, δίνεται το γράφημα που δείχνει την σχέση μεταξύ του MSE και του αριθμού των bits της κωδικοποιημένης εικόνας.



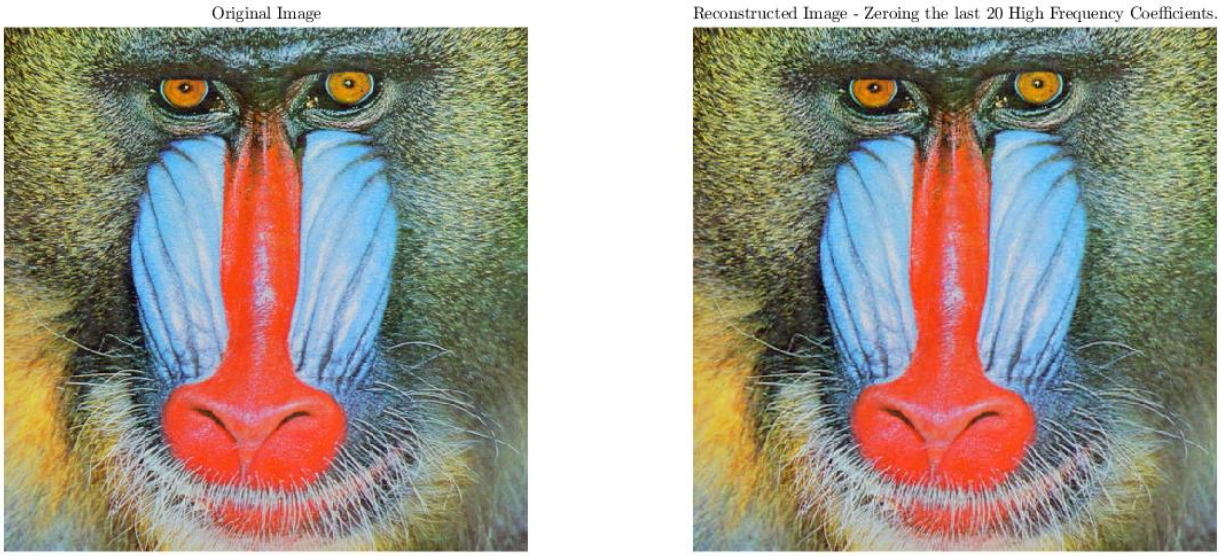
Εικόνα 28: Αριθμός bits και MSE [Δεύτερη Εικόνα]

Η διαίσθηση μας, πως η αύξηση του αριθμού των bits θα οδηγήσει σε μείωση του MSE επιβεβαιώνεται.

Μηδενισμός υψίσυχνων όρων DCT – Πρώτη εικόνα

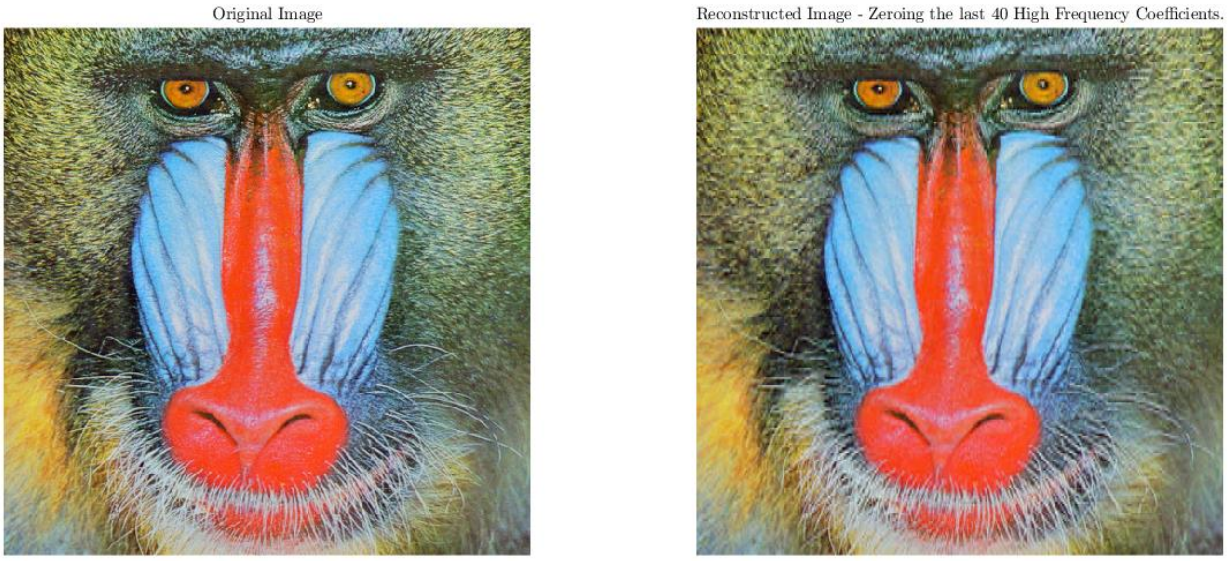
Για τον μηδενισμό των k υψίσυχνων όρων, πρέπει να τροποποιήσουμε το περιεχόμενο των k τελευταίων στοιχείων (σε Zig-Zag μοτίβο) των πινάκων κβαντισμού. Έτσι, αντικαθιστούμε τις τιμές τους με τη τιμή 1000 (θέλουμε μεγάλο διαιρέτη).

- <u>Μηδενισμός 20 υψίσυχνων όρων DCT</u>



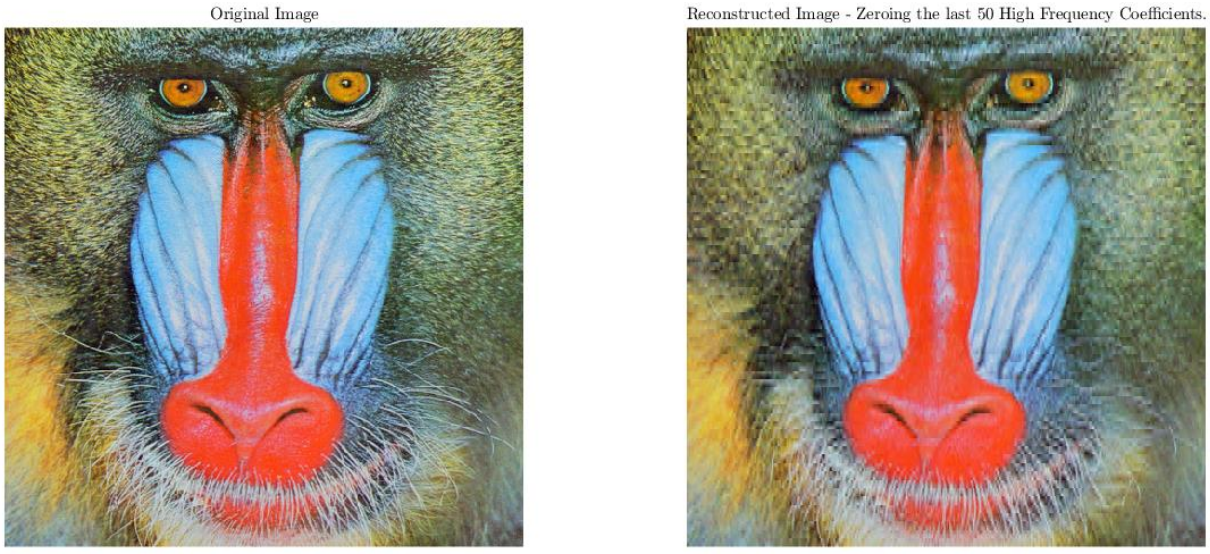
Εικόνα 29: Πρώτη εικόνα - Υποδειγματοληψία $4{:}4{:}4$*,* $qScale = 1$*, μηδενισμός 20 υψίσυχνων όρων [demo Results]*

- Μηδενισμός 40 υψίσυχνων όρων DCT



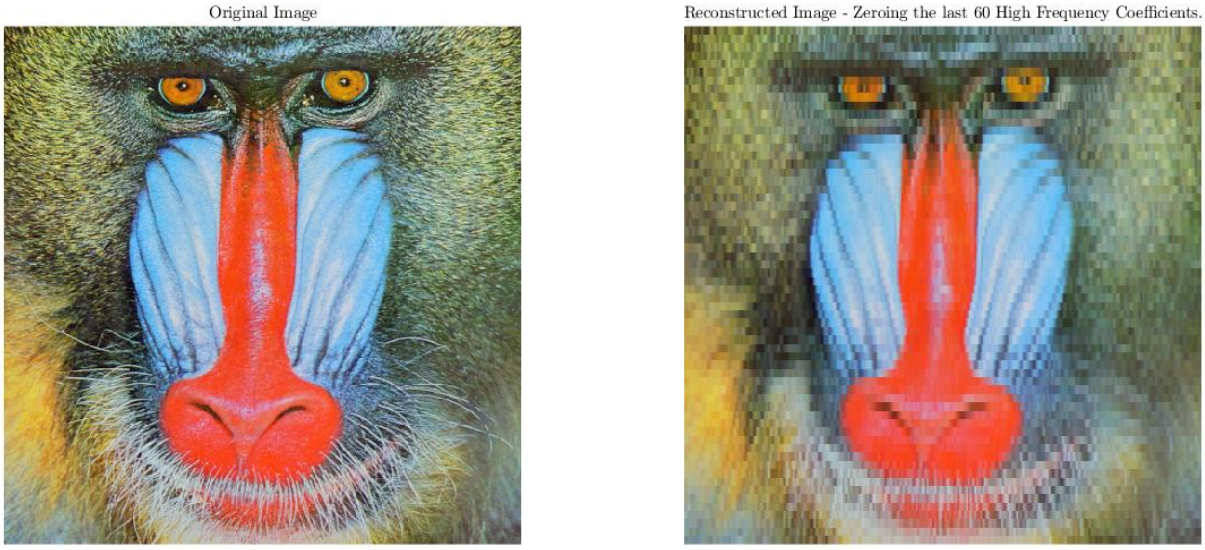
Εικόνα 30: Πρώτη εικόνα - Υποδειγματοληψία $4:4:4$*,* $qScale = 1$*, μηδενισμός 40 υψίσυχνων όρων [demo Results]*

- Μηδενισμός 50 υψίσυχνων όρων DCT



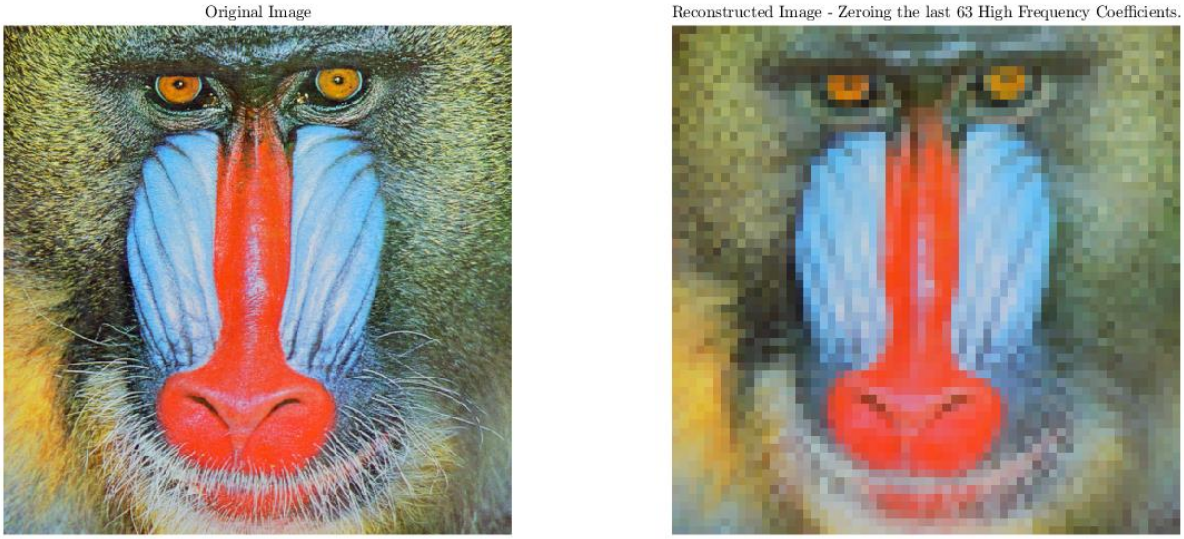
Εικόνα 31: Πρώτη εικόνα - Υποδειγματοληψία $4:4:4$*,* $qScale = 1$*, μηδενισμός 50 υψίσυχνων όρων [demo Results]*

- Μηδενισμός 60 υψίσυχνων όρων DCT



Εικόνα 32: Πρώτη εικόνα - Υποδειγματοληψία $4{:}\,4{:}\,4$*,* $qScale = 1$*, μηδενισμός 60 υψίσυχνων όρων [demo Results]*

- Μηδενισμός 63 υψίσυχνων όρων DCT



Εικόνα 33: Πρώτη εικόνα - Υποδειγματοληψία $4{:}\,4{:}\,4$*,* $qScale = 1$*, μηδενισμός 63 υψίσυχνων όρων [demo Results]*

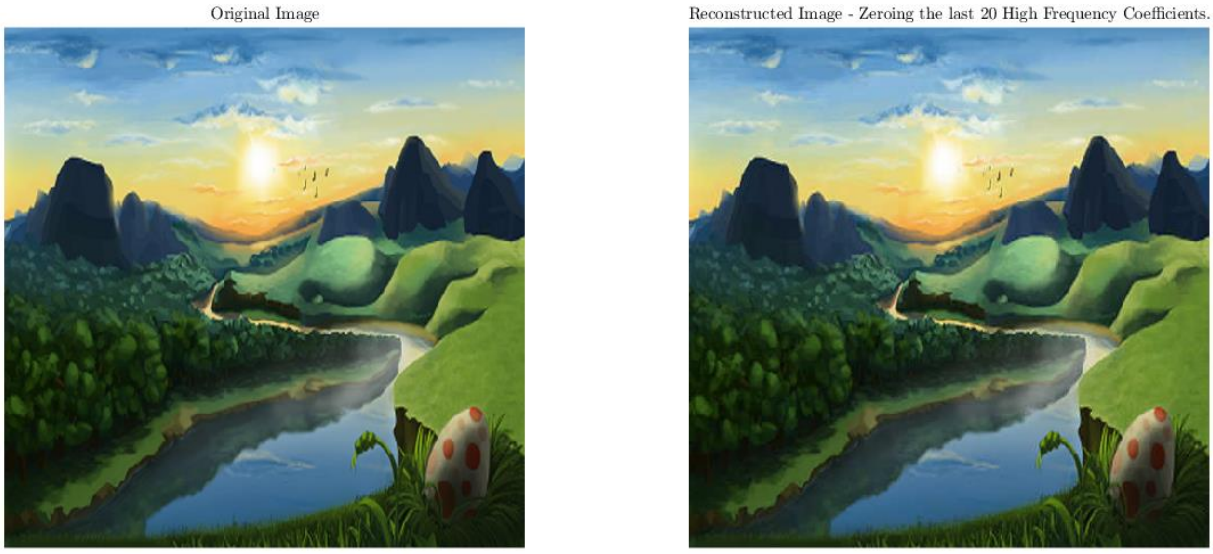
Όσο περισσότερους όρους μηδενίσουμε, τόσο λιγότερη λεπτομέρεια περνάει στην τελική μορφή της εικόνας. Ωστόσο, παρατηρούμε πως ακόμα και αν μηδενίσουμε 63 όρους (όλους τους AC όρους!), μόνο η πληροφορία του DC όρου, αρκεί για να δωθεί η γενικότερη μορφή που περιέχεται στην εικόνα.

Σημείωση: Δεν δόθηκε διευκρίνηση για τον τύπο της υποδειγματοληψίας, έτσι χρησιμοποιήθηκε η $4{:}\,4{:}\,4$ (με $qScale = 1$), για να περιοριστούν οι πηγές εισαγωγής σφάλματος και να γίνει μια εκτίμηση του πόσου επηρεάζουν οι μηδενισμοί υψίσυχνων όρων DCT το τελικό αποτέλεσμα.

Μηδενισμός υψίσυχνων όρων DCT – Δεύτερη εικόνα

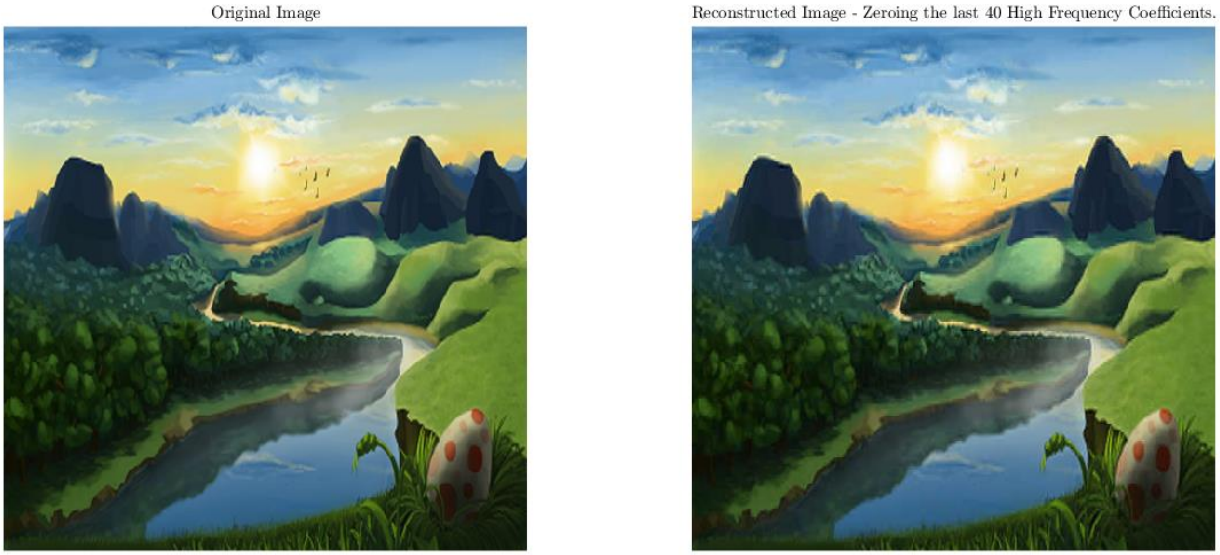
Η ίδια διαδικασία επαναλαμβάνεται για την δεύτερη εικόνα.

- Μηδενισμός 20 υψίσυχνων όρων DCT



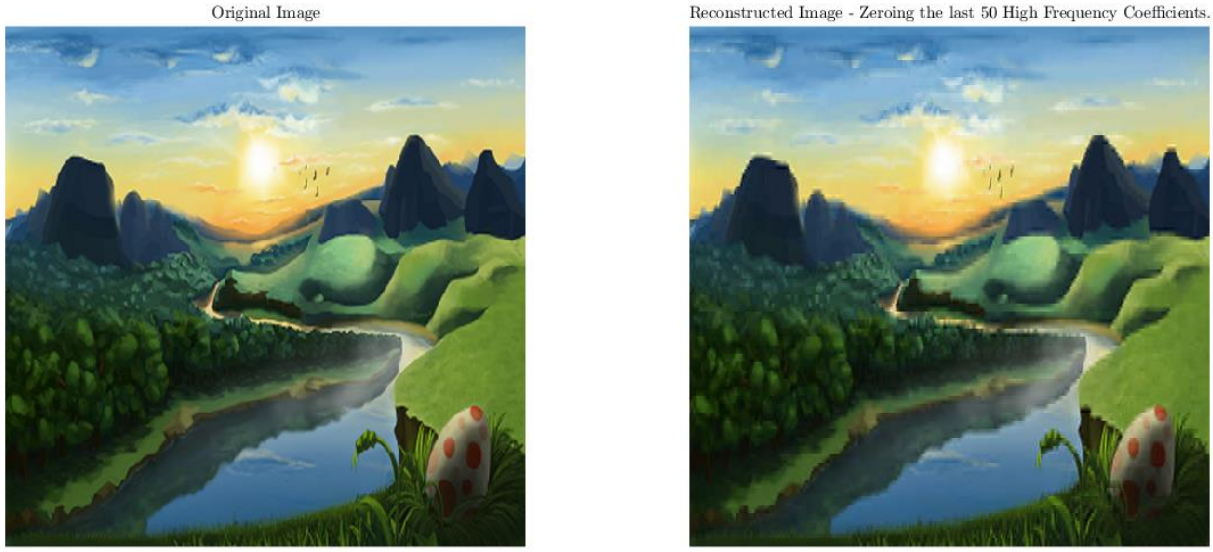
Εικόνα 34: Δεύτερη εικόνα - Υποδειγματοληψία $4{:}4{:}4$*,* $qScale = 1$*, μηδενισμός 20 υψίσυχνων όρων [demo Results]*

- Μηδενισμός 40 υψίσυχνων όρων DCT



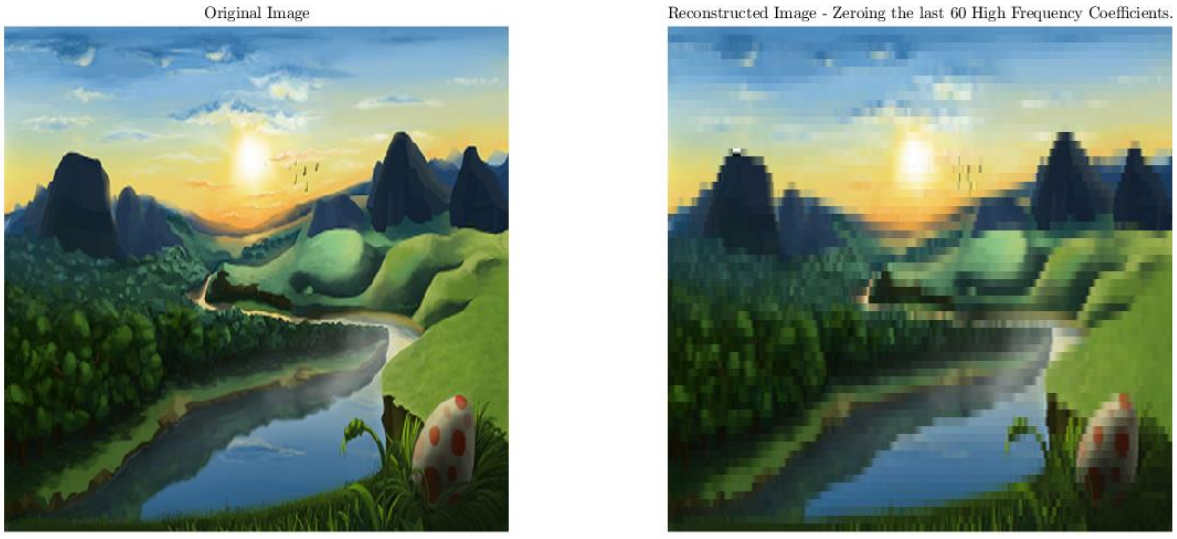
Εικόνα 35: Δεύτερη εικόνα - Υποδειγματοληψία $4{:}4{:}4$*,* $qScale = 1$*, μηδενισμός 40 υψίσυχνων όρων [demo Results]*

- Μηδενισμός 50 υψίσυχνων όρων DCT



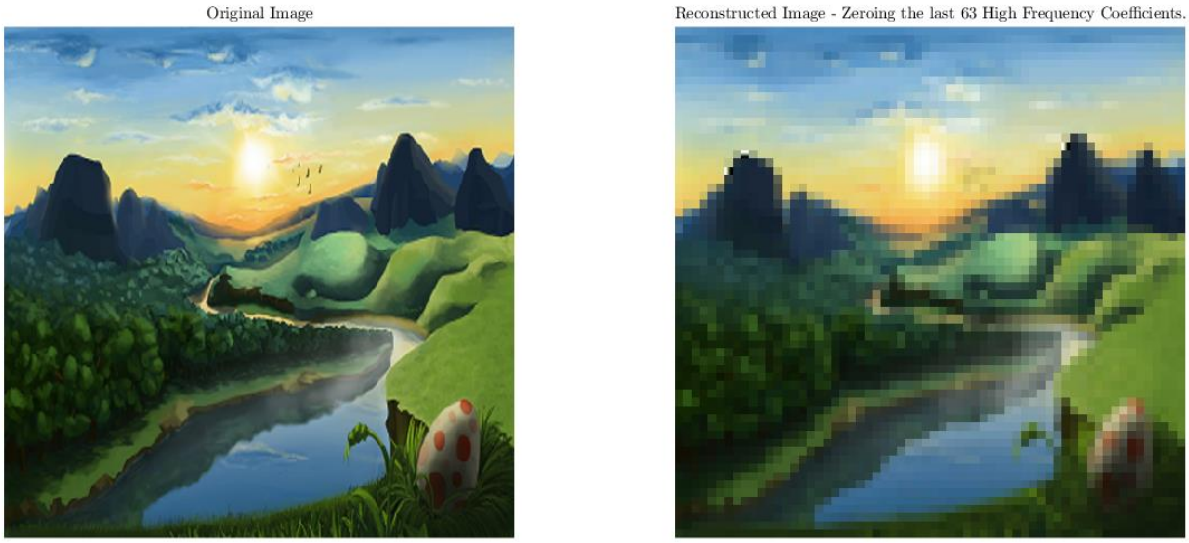
Εικόνα 36: Δεύτερη εικόνα - Υποδειγματοληψία $4{:}4{:}4$*,* $qScale = 1$*, μηδενισμός 50 υψίσυχνων όρων [demo Results]*

- Μηδενισμός 60 υψίσυχνων όρων DCT



Εικόνα 37: Δεύτερη εικόνα - Υποδειγματοληψία $4{:}4{:}4$*,* $qScale = 1$*, μηδενισμός 60 υψίσυχνων όρων [demo Results]*

- Μηδενισμός 63 υψίσυχνων όρων DCT



Εικόνα 38: Δεύτερη εικόνα - Υποδειγματοληψία $4{:}\,4{:}\,4$*,* $qScale = 1$*, μηδενισμός 63 υψίσυχνων όρων [demo Results]*

Τα συμπεράσματα και με αυτήν την εικόνα είναι τα ίδια με πριν. Επιβεβαιώνεται έτσι για τον μετασχηματισμό DCT η εξαιρετική ικανότητα συγκέντρωσης της πληροφορίας σε ένα μικρό πλήθος δειγμάτων.

Σημείωση: Δεν δόθηκε διευκρίνηση για τον τύπο της υποδειγματοληψίας, έτσι χρησιμοποιήθηκε η $4{:}\,4{:}\,4$ (με $qScale = 1$), για να περιοριστούν οι πηγές εισαγωγής σφάλματος και να γίνει μια εκτίμηση του πόσου επηρεάζουν οι μηδενισμοί υψίσυχνων όρων DCT το τελικό αποτέλεσμα.

Υπολογισμός εντροπίας – Πρώτη εικόνα

Αναφέρθηκε και προηγουμένως πως ο μετασχηματισμός DCT, χρησιμοποιείται για την αποσυσχέτιση των δεδομένων. Μια τέτοια διαδικασία θα μειώσει την εντροπία ανά σύμβολο του συστήματος και θα οδηγήσει σε καλύτερη συμπίεση. Επιπλέον, κατά τον υπολογισμό των μηκών διαδρομής ευελπιστούμε σε μείωση των υπό προς κωδικοποίηση συμβόλων, με σκοπό την μείωση της συνολικής εντροπίας της εικόνας. Για την εντροπία ισχύει

$$Entropy = \sum_i -f_i * \log_2(f_i) \ [per\ symbol]$$

Παρακάτω γίνεται σύγκριση μεταξύ της εντροπίας ανά σύμβολο και της συνολικής εντροπίας της εικόνας σε τρία στάδια της κωδικοποίησης.

Spatial Domain – RGB: Η εντροπία υπολογίζεται ως το άθροισμα των εντροπιών των καναλιών R, G και B. Για να ισχύει κάτι τέτοιο, πρέπει να γίνει η παραδοχή ότι οι πιθανότητες εμφάνισης των τιμών είναι ανεξάρτητες για κάθε κανάλι χρώματος[5]. Τότε,

$$Image\ Entropy = EntropyRed * length(Red) + EntropyGreen * length(Green) + EntropyBlue * length(Blue)$$

[5] Δηλαδή για μια τριάδα τιμών [r g b], ισχύει $p(x,y,z) = p(x)p(y)p(z)$.

Quantized DCT Coefficients: Όμοια με την παραπάνω περίπτωση.

$$Image\ Entropy = EntropyY * length(Y) + EntropyCb * length(Cb) + EntropyCr * length(Cr)$$

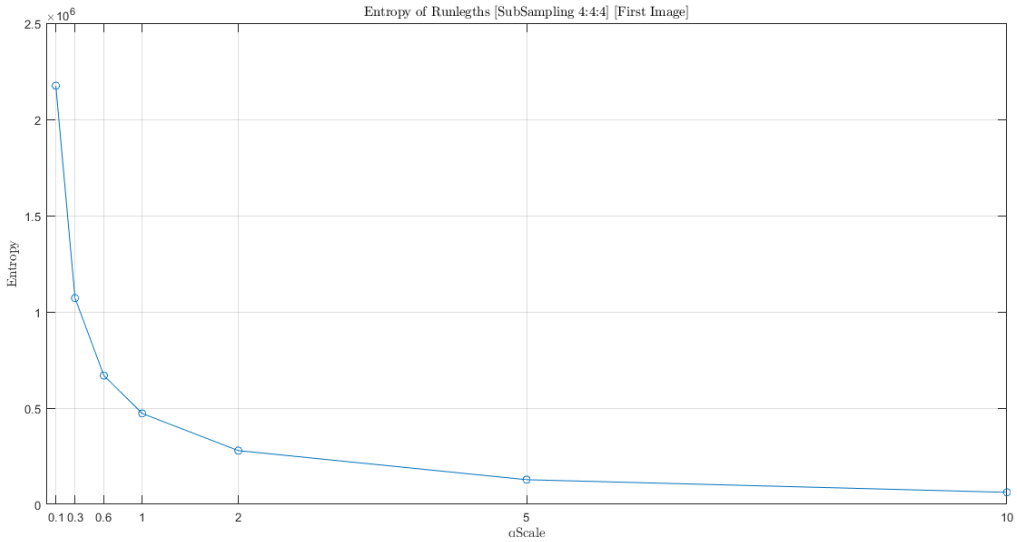
Runlengths: Υπολογίζεται η συχνότητα εμφάνισης του κάθε ζεύγους τιμών. Στην συνέχεια γίνεται χρήση του τύπου της εντροπίας.

$$Image\ Entropy = Entropy * length(Runlenghts)$$

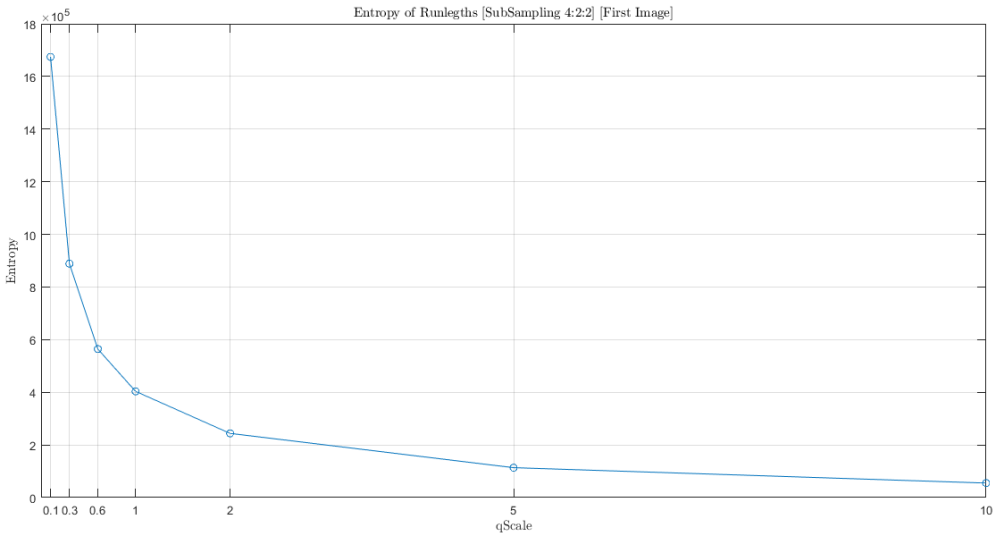
Στην συνέχεια παρουσιάζονται οι εντροπίες υπολογισμένες για την πρώτη εικόνα, με $qScale = 0.6$ και υποδειγματοληψία 4: 2: 2, όπου η συνολική εντροπία μειώνεται, που ήταν και το ζητούμενο.

	Εντροπία	
	per Symbol	**Συνολική**
Spatial Domain – RGB	22.897278	$5.633097 * 10^6$
Quantized DCT Coefficients	3.670057	$7.255579 * 10^5$
Runlengths	5.055575	$5.643539 * 10^5$

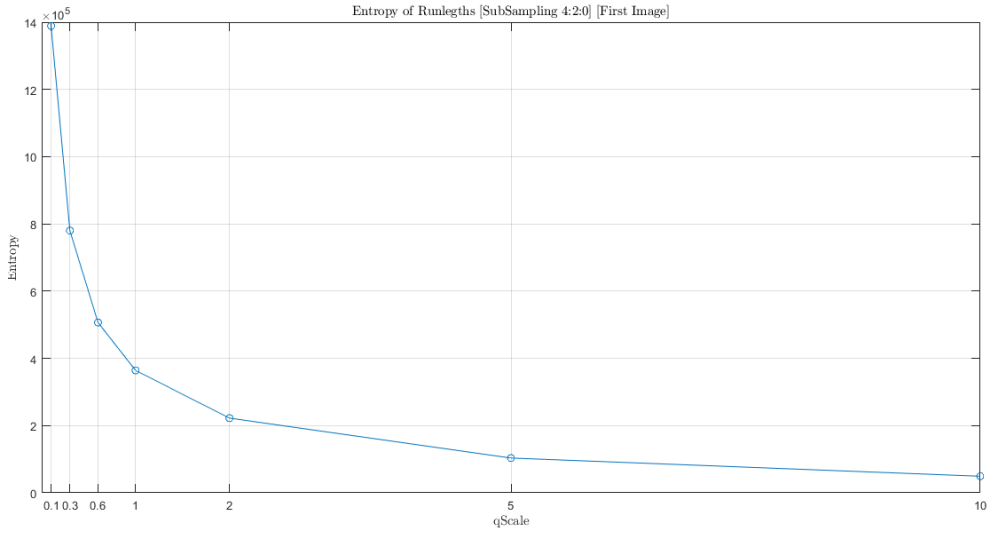
Για να παρατηρηθεί η σχέση μεταξύ υποδειγματοληψίας, $qScale$ και συνολικής εντροπίας, παρακάτω παρουσιάζονται για τα Runlengths της πρώτης εικόνα οι διάφοροι συνδυασμοί υποδειγματοληψίας και $qScale$.



Εικόνα 39: Συνολική εντροπία Runlengths και $qScale$ - Υποδειγματοληψία 4: 4: 4 [Πρώτη εικόνα]



Εικόνα 40: Συνολική εντροπία Runlengths και $qScale$ - Υποδειγματοληψία $4:2:2$ [Πρώτη εικόνα]



Εικόνα 41: Συνολική εντροπία Runlengths και $qScale$ - Υποδειγματοληψία $4:2:0$ [Πρώτη εικόνα]

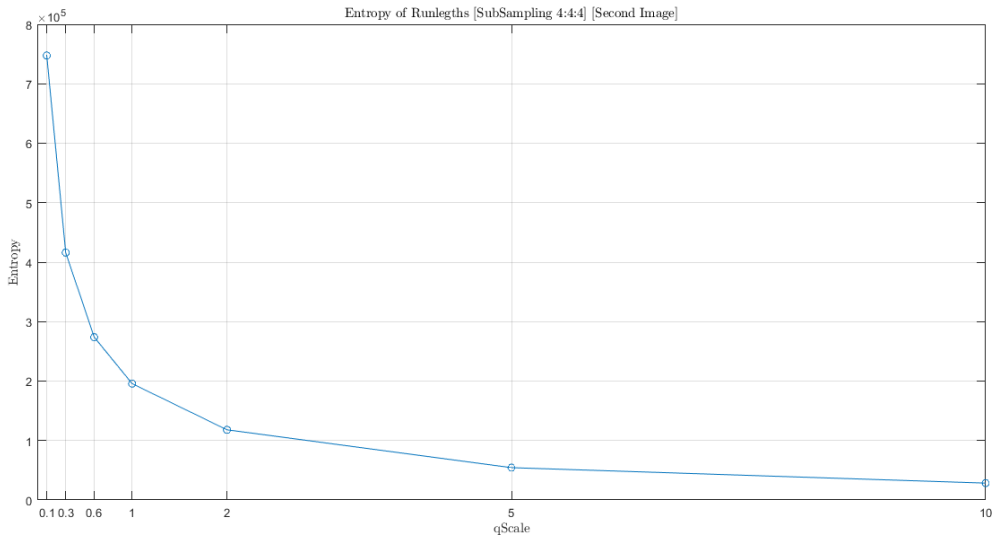
Είναι φανερό πως η συνολική εντροπία αφενός μειώνεται κατά την αύξηση του $qScale$ και αφετέρου για το ίδιο $qScale$ η εντροπία είναι μικρότερη για μεγαλύτερες υποδειγματοληψίες.

Υπολογισμός εντροπίας – Δεύτερη εικόνα

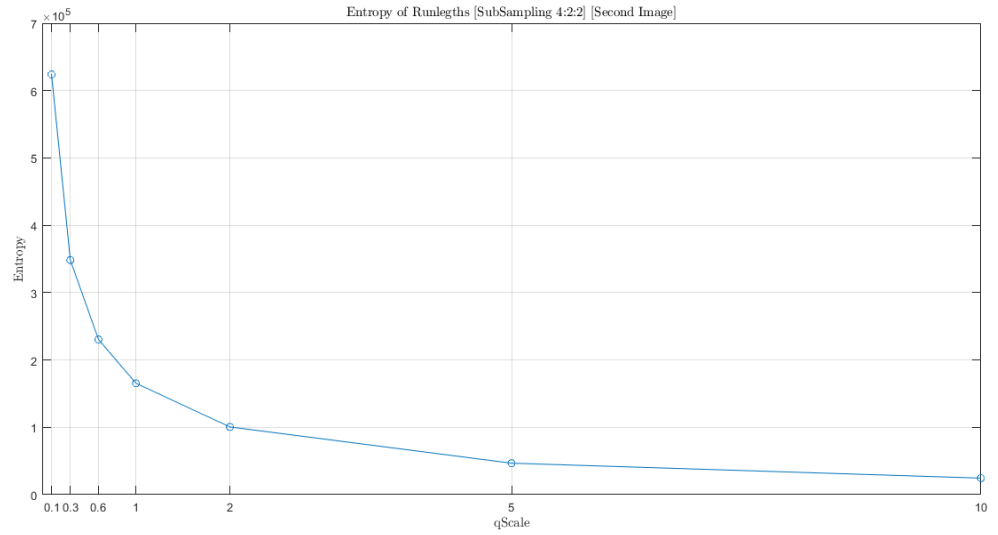
Ακολουθώντας το ίδιο σκεπτικό για την δεύτερη εικόνα, με $qScale = 5$ και υποδειγματοληψία $4:4:4$.

	Εντροπία	
	per Symbol	**Συνολική**
Spatial Domain – RGB	22.915565	$5.637596 * 10^6$
Quantized DCT Coefficients	0.604404	$1.486930 * 10^5$
Runlengths	1.933115	$5.439012 * 10^4$

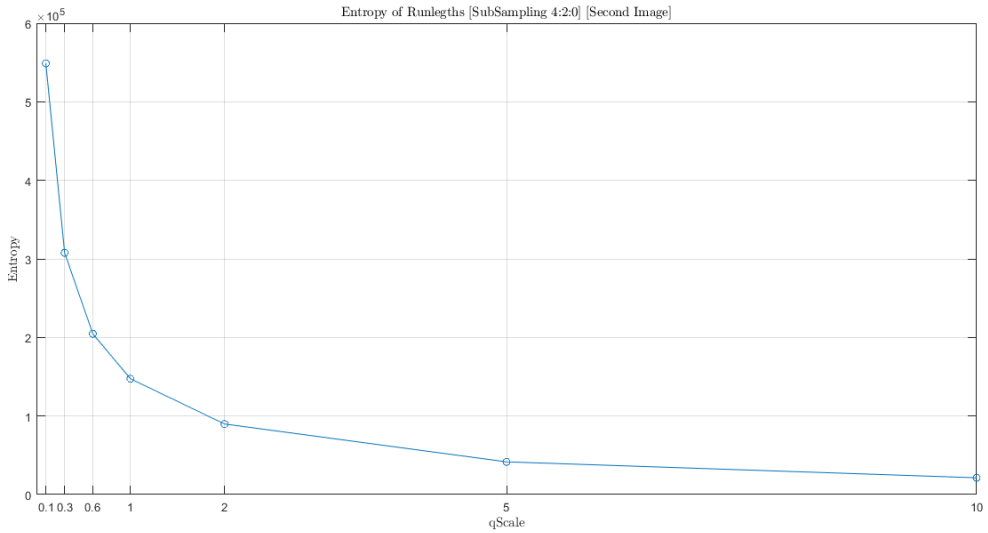
Ομοίως γίνεται η ανάλυση για την δεύτερη εικόνα.



Εικόνα 42: Συνολική εντροπία Runlengths και qScale - Υποδειγματοληψία 4: 4: 4 [Δεύτερη εικόνα]



Εικόνα 43: Συνολική εντροπία Runlengths και qScale - Υποδειγματοληψία 4: 2: 2 [Δεύτερη εικόνα]



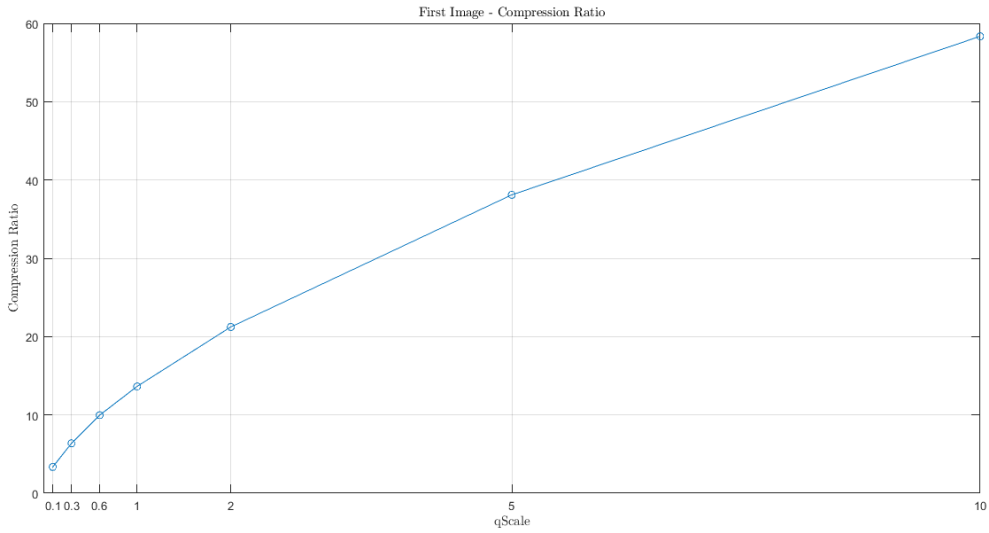
Εικόνα 44: Συνολική εντροπία Runlengths και $qScale$ - Υποδειγματοληψία $4:2:0$ [Δεύτερη εικόνα]

Είναι φανερό και σε αυτή την περίπτωση, πως η συνολική εντροπία αφενός μειώνεται κατά την αύξηση του $qScale$ και αφετέρου για το ίδιο $qScale$ η εντροπία είναι μικρότερη για μεγαλύτερες υποδειγματοληψίες.

Compression ratio – Πρώτη εικόνα

Για να υπολογιστεί ο λόγος συμπίεσης, αρκεί να μετρήσουμε το πλήθος των στοιχείων της αρχικής εικόνας και να το συγκρίνουμε με το συνολικό μέγεθος των huffStream. Για την πρώτη εικόνα με υποδειγματοληψία $4:2:2$ και διάφορες τιμές του $qScale$, τα αποτελέσματα δίνονται παρακάτω

	$qScale$						
	0.1	*0.3*	*0.6*	*1*	*2*	*5*	*10*
Compression Ratio	3.3543	6.3703	9.9626	13.6367	21.2290	38.1026	58.3483

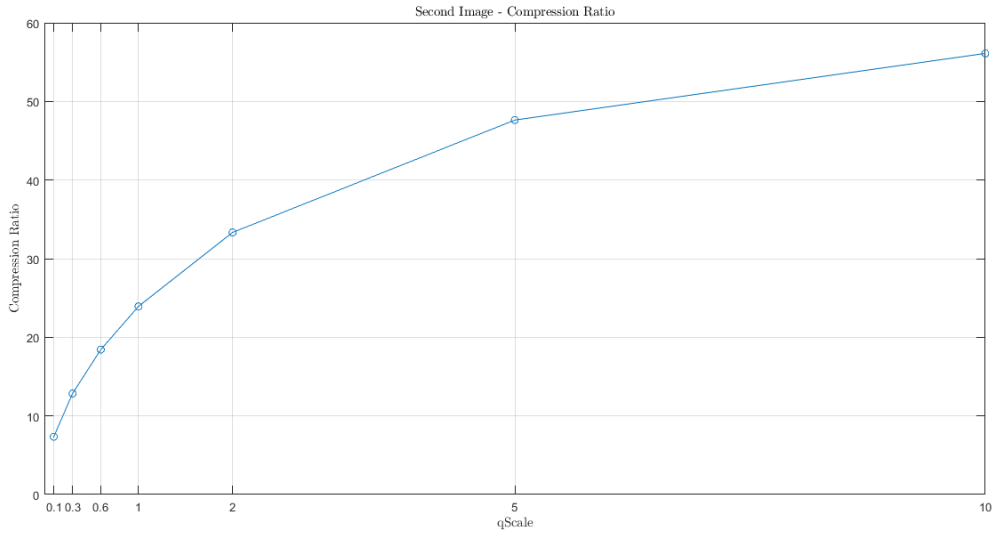


Εικόνα 45: Compression Rate και $qScale$ [Πρώτη Εικόνα]

Compression ratio – Δεύτερη εικόνα

Με την ίδια μεθοδολογία, για την δεύτερη εικόνα με υποδειγματοληψία $4{:}\,4{:}\,4$ και διάφορες τιμές του $qScale$, τα αποτελέσματα δίνονται παρακάτω

	$qScale$						
	0.1	*0.3*	*0.6*	*1*	*2*	*5*	*10*
Compression Ratio	7.3489	12.8528	18.4374	23.9269	33.3370	47.6314	56.1083



Εικόνα 46: Compression Rate και $qScale$ [Δεύτερη Εικόνα]

Παρατηρούμε, πως όσο αυξάνει το $qScale$ τόσο καλύτερο λόγο συμπίεσης επιτυγχάνουμε. Βέβαια, είδαμε και προηγουμένως πως αύξηση του $qScale$, οδηγεί κι σε μεγαλύτερα σφάλματα. Τέλος, τα παραπάνω αποτελέσματα δεν αποτυπώνουν τον λόγο συμπίεσης για το πραγματικό μέγεθος της κωδικοποιημένης εικόνας, καθώς λείπουν στοιχεία κρίσιμα για την επιτυχημένη αποκωδικοποίηση της από κάποιο πρόγραμμα επεξεργασίας εικόνων (πχ. headers), ωστόσο αποδεικνύει πως τα διάφορα στάδια του προτύπου οδηγούν σε μείωση του μεγέθους της κωδικοποιημένης εικόνας.