Αναφορά Εργασίας

Συστήματα Πολυμέσων

Ιανουάριος – Φεβρουάριος 2024

Χριστοδούλου Βάιος

9987

vkchrist@ece.auth.gr

Περιεχόμενα

Εισαγωγή

Η παρούσα αναφορά αποτελεί κομμάτι της εργασίας μου για το μάθημα Συστήματα Πολυμέσων, όπως αυτό διδάχτηκε το χειμερινό ακαδημαϊκό εξάμηνο 2023-2024. Ο πλήρης κώδικας βρίσκεται ανεβασμένος στο GitHub. Στόχος της εργασίας ήταν η υλοποίηση ενός κωδικοποιητή και ενός αποκωδικοποιητή σύμφωνα με το πρότυπο JPEG.

Σημειώνεται ότι ενώ ακολουθήθηκε η οδηγία περί απουσίας φακέλων στο .zip παραδοτέο αρχείο, κρατήθηκαν ένας φάκελος για τις εικόνες και ένας φάκελος που περιέχει τα αρχείο των πινάκων (κβαντισμού, Huffman), με σκοπό την καλύτερη οργάνωση.

1ο Παραδοτέο

Στο πρώτο παραδοτέο, καλούμαστε να υλοποιήσουμε τις αρχικές συναρτήσεις μετασχηματισμού και κβαντισμού. Πιο συγκεκριμένα, ζητούνται ο μετασχηματισμός στο χρωματικό σύστημα YCrCb, ο διακριτός μετασχηματισμός συνημιτόνων (DCT) και ο κβαντισμός των DCT συντελεστών. Εκτός αυτών, ζητείται και η αντίστροφη διαδικασία: η μετατροπή των κβαντισμένων DCT συντελεστών πίσω σε RGB συνιστώσες, και η ανακατασκευή της εικόνας.

YCrCb

Το πρώτο στάδιο είναι η μετατροπή των τριών χρωματικών συνιστωσών, Red (R), Green (G), Blue (B), σε αυτές του χρωματικού συστήματος YCrCb, και το αντίστροφο. Το αρχείο που αναλαμβάνει αυτήν την δουλειά, είναι το rgb_to_ycrcb.py. Η διαδικασία αυτή είναι τετριμμένη, υλοποιείται μέσω της συνάρτησης convert2ycrcb, και τελείται μέσω ενός πίνακα μετασχηματισμού, ο οποίος φαίνεται παρακάτω:

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

Η αντίστροφη διαδικασία, η μετατροπή, δηλαδή, των συνιστωσών του YCrCb χρωματικού συστήματος, στις βασικές χρωματικές συνιστώσες, είναι προφανώς ο πολλαπλασιασμός με τον αντίστροφο πίνακα μετασχηματισμού:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 1.402 & 0 \\ 1 & -0.7141 & -0.3441 \\ 1 & 0 & 1.772 \end{bmatrix} \begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix}$$

Υποδειγματοληψία

Η υποδειγματοληψία συνεισφέρει στην μείωση της πληροφορίας, καθώς ελαττώνει το πλήθος των συνιστωσών Cr και Cb που αποθηκεύονται. Οι ζητούμενες υλοποιήσεις αφορούν τις υποδειγματοληψίες 4:4:4, 4:2:2 και 4:2:0. Για τον σκοπό αυτό υλοποιήθηκε η συνάρτηση subsampleCrCb. Κατά την αντίστροφη διαδικασία, κρίνεται απαραίτητη μια διαδικασία υπερδειγματοληψίας, προκειμένου να επιστρέψει το πλήθος των συνιστωσών Cr και Cb στο αναμενόμενο – αρχικό. Έτσι, υλοποιήθηκε η συνάρτηση upsampleCrCb, η οποία πραγματοποιεί μια γραμμική παρεμβολή μεταξύ των συνιστωσών, για να διπλασιαστεί το πλήθος τους, στους άξονες x και y, όπως αυτό κρίνεται από το είδος της δειγματοληψίας. Οι συναρτήσεις intepolateRows και interpolateColumns (για τους άξονες x και y, αντίστοιχα), έχουν αυτόν ακριβώς τον σκοπό.

Μετασχηματισμός DCT

Οι προηγούμενες συναρτήσεις λειτουργούν σε επίπεδο εικόνας. Προκειμένου, όμως, να εφαρμοστεί το πρότυπο JPEG, οι επόμενες συναρτήσεις λειτουργούν σε επίπεδο block, όπου ως block ορίζουμε κάθε 8x8 πίνακα στοιχείων. Ο πρώτος μετασχηματισμός τέτοιου είδους είναι ο διακριτός μετασχηματισμός συνημιτόνων (DCT). Αυτός, καθώς και ο αντίστροφός του, υλοποιούνται στο αρχείο dct.py. Η υλοποίηση είναι εξαιρετικά απλή, μιας και υπάρχουν έτοιμες μαθηματικές βιβλιοθήκες που αναλαμβάνουν τους μαθηματικούς υπολογισμούς, όπως η scipy. Ο ευθύς μετασχηματισμός, μετατρέπει τα 64 (8x8) στοιχεία κάθε block, σε μία DC συνιστώσα και 63 AC, όπως αποκαλούνται.

Κβαντισμός Συντελεστών

Κατά τον κβαντισμό των DCT συντελεστών, αυτοί στρογγυλοποιούνται, σύμφωνα με έναν πίνακα και μια κλίμακα κβαντισμού: qTable και qScale αντίστοιχα. Οι συναρτήσεις αυτές, οι οποίες είναι επίσης απλές, υλοποιούνται στο αρχείο quantization.py. Ο μαθηματικός τύπος που δίνει τον κβαντισμένο συντελεστή *i, j* είναι ο παρακάτω:

$$q[i][j] = round(\frac{DCT[i][j]}{(qTable[i][j] * qScale)})$$

Προφανώς, ο αποκβαντισμένος συντελεστής *i, j* δίνεται από τον τύπο:

$$DCT[i][j] = q[i][j] * qTable[i][j] * qScale$$

Ο πίνακας qTable είναι διαφορετικός για την συνιστώσα Y και διαφορετικός για τις Cr και Cb, δίνεται από το πρότυπο και υπάρχει στο αρχείο quantization_tables.py. Αυτός είναι ο πρώτος μετασχηματισμός κατά τον οποίο παρατηρείται μείωση πληροφορίας, εξαιτίας της στρογγυλοποίησης των DCT συντελεστών.

Demo 1

Το τελικό ζητούμενο του πρώτου παραδοτέο είναι ένα αρχείο που να επιδεικνύει την λειτουργικότητα των προηγούμενων μετασχηματισμών, το demo1.py, πάνω σε δύο δοθείσες εικόνες. Οι διαδικασίες που πρέπει να γίνουν είναι δύο:

1. Η μετατροπή των R, G, B καναλιών των εικόνων στο σύστημα YCrCb, σύμφωνα με υποδειγματοληψίες 4:2:2 και 4:4:4 για τις εικόνες 1 και 2 αντίστοιχα. Έπειτα, πρέπει να γίνει η αντίστροφη μετατροπή, στο σύστημα RGB και η ανακατασκευή των εικόνων. Οι εικόνες εμφανίζονται και έπειτα αποθηκεύονται στον φάκελο images, με ονόματα baboon1.png και lena1.png, αντίστοιχα. Την διαδικασία αυτή εκτελεί η συνάρτηση question1.
2. Η μετατροπή του συστήματος RGB στο YCrCb, όπως παραπάνω, ο υπολογισμός των DCT συντελεστών και ο κβαντισμός τους, με τιμές qScale = 0.6 και qScale = 5, για τις εικόνες 1

και 2 αντίστοιχα. Έπειτα, πρέπει να γίνει η αντίστροφη μετατροπή και η ανακατασκευή των εικόνων, όπως παραπάνω. Οι εικόνες εμφανίζονται και έπειτα αποθηκεύονται, με ονόματα baboon2.png και lena2.png, αντίστοιχα. Την διαδικασία αυτή εκτελεί η συνάρτηση question2.

Αποτελέσματα

Οι ανακατασκευασμένες εικόνες φαίνονται παρακάτω.

Εικόνα 1. Ανακατασκευασμένες εικόνες από την διαδικασία 1

Εικόνα 2. Ανακατασκευασμένες εικόνες από την διαδικασία 2

Είναι εύκολο να παρατηρηθεί ότι οι δύο πρώτες εικόνες δεν παρουσιάζουν διαφορές από τις δοθείσες, τουλάχιστον με γυμνό μάτι. Επίσης, παρατηρείται ότι η δεύτερη εικόνα της δεύτερης

διαδικασίας παρουσιάζει μειωμένη ευκρίνεια σε σχέση με την αρχική, φαινόμενο αναμενόμενο, δεδομένου της αυξημένης κλίμακας κβαντισμού.

2ο Παραδοτέο

Στο δεύτερο παραδοτέο ζητείται η υλοποίηση των επόμενων σταδίων της κωδικοποίησης μιας εικόνας, σύμφωνα με το πρότυπο JPEG. Αυτά είναι η κωδικοποίηση μήκους-διαδρομής (run-length encoding, RLE) και η κωδικοποίηση Huffman. Προφανώς, όπως και προηγουμένως, ζητούνται και οι αντίστροφοι μετασχηματισμοί. Τέλος, ζητούνται οι γενικές συναρτήσεις κωδικοποίησης και αποκωδικοποίησης JPEG, που θα ενσωματώνουν τους παραπάνω μετασχηματισμούς.

Κωδικοποίηση Μήκους-Διαδρομής

Η κωδικοποίηση μήκους-διαδρομής υλοποιείται στο αρχείο zig_zag.py, κατά την οποία συντελούνται δύο στάδια.

Σκανάρισμα Ζιγκ-Ζαγκ

Αρχικά, οι κβαντισμένοι DCT συντελεστές του εκάστοτε block, μετατρέπονται σε ένα (μονοδιάστατο) διάνυσμα, το οποίο υπολογίζεται από ένα ζιγκ-ζαγκ σκανάρισμα (zig-zag scanning) του αρχικού block. Αυτό βοηθάει στο να μεταφερθούν οι μη μηδενικοί συντελεστές στην αρχή και οι μηδενικοί στο τέλος του διανύσματος. Η ακριβής σειρά σκαναρίσματος ορίζεται στο πρότυπο.

Κωδικοποίηση Μήκους-Διαδρομής

Έπειτα, εφαρμόζεται μια μέθοδος η οποία βοηθάει στην μείωση του μεγέθους, χωρίς μείωση της πληροφορίας, υπό την προϋπόθεση ότι εμφανίζονται συνεχόμενα μηδενικά. Το τελικό διάνυσμα που επιστρέφεται, αποτελείται από δυάδες (σύμβολα), εκ των οποίων το πρώτο στοιχείο δηλώνει τον αριθμό των μηδενικών συντελεστών πριν από κάποιον μη μηδενικό, και το δεύτερο τον μη μηδενικό συντελεστή. Σημειώνεται ότι οι DC συντελεστές κωδικοποιούνται με διαφορετικό τρόπο, καθώς είναι μοναδικοί για κάθε block και δεν θα είχε νόημα η παραπάνω κωδικοποίηση. Για αυτούς, οι οποίοι αποτελούν το πρώτο σύμβολο του διανύσματος, το πρώτο στοιχείο της συμβόλου είναι πάντα μηδέν, και το δεύτερο είναι η διαφορά μεταξύ των DC συντελεστών του εν λόγω και του προηγούμενου block. Έτσι, στην συχνή περίπτωση που δύο συνεχόμενα block έχουν παρόμοιο DC συντελεστή, η τιμή που κωδικοποιείται είναι μικρή.

Στην αντίστροφη διαδικασία υπολογίζεται το block το οποίο έδωσε το διάνυσμα μήκους-διαδρομής, και, έπειτα, μέσω του αντιστρόφου σκαναρίσματος, υπολογίζεται το block των κβαντισμένων DCT συντελεστών.

Κωδικοποίηση Huffman

Το τελευταίο στάδιο της κωδικοποίησης κατά το πρότυπο JPEG, αποτελεί ο μετασχηματισμός των συμβόλων μήκους-διαδρομής σε κάποιο bitstream. Ενώ υπάρχουν διάφορες μέθοδοι για να επιτευχθεί αυτό, εμείς καλούμαστε να υλοποιήσουμε την μέθοδο Huffman, η οποία είναι και η βέλτιστη. Το αρχείο που επιτελεί αυτόν τον σκοπό είναι το huffman.py.

Η εν λόγω κωδικοποίηση, όπως ήταν αναμενόμενο, είναι διαφορετική για τους DC και τους AC συντελεστές του διανύσματος συμβόλων.

DC Συντελεστές

Μέσω πινάκων, υπολογίζεται η κατηγορία στην οποία ανήκει η διαφορά των δύο διαδοχικών συντελεστών (DIFF), στην οποία αντιστοιχεί και κάποιο bitstream. Έπειτα, προκειμένου να ανιχνευθεί το συγκεκριμένο σύμβολο της κατηγορίας που κωδικοποιείται, προστίθενται κάποια επιπλέον bit. Αν το DIFF είναι θετικό, προστίθεται απλώς ο αριθμός στο δυαδικό σύστημα. Αν το DIFF είναι αρνητικό, προστίθεται το συμπλήρωμα ως προς το 2 της απόλυτης τιμής του DIFF-1 στο δυαδικό σύστημα. Με αυτόν τον τρόπο, καταφέρνουμε το πρώτο bit του προστιθέμενου bitstream να είναι 1 για θετικά και 0 για αρνητικά DIFF.

AC Συντελεστές

Για τους AC συντελεστές ακολουθείται παρόμοια διαδικασία. Υπολογίζεται η κατηγορία του συμβόλου, και έπειτα υπάρχουν πίνακες που αντιστοιχίζουν την δυάδα *(αριθμός συνεχόμενων μηδενικών, κατηγορία)* σε κάποιο bitstream. Έπειτα, για τον ίδιο λόγο με πριν, προστίθενται ορισμένα bit. Ο υπολογισμός τους γίνεται με τον ίδιο τρόπο, απλώς αντί για την διαφορά δύο διαδοχικών συντελεστών, χρησιμοποιείται η τιμή του μη μηδενικού συμβόλου.

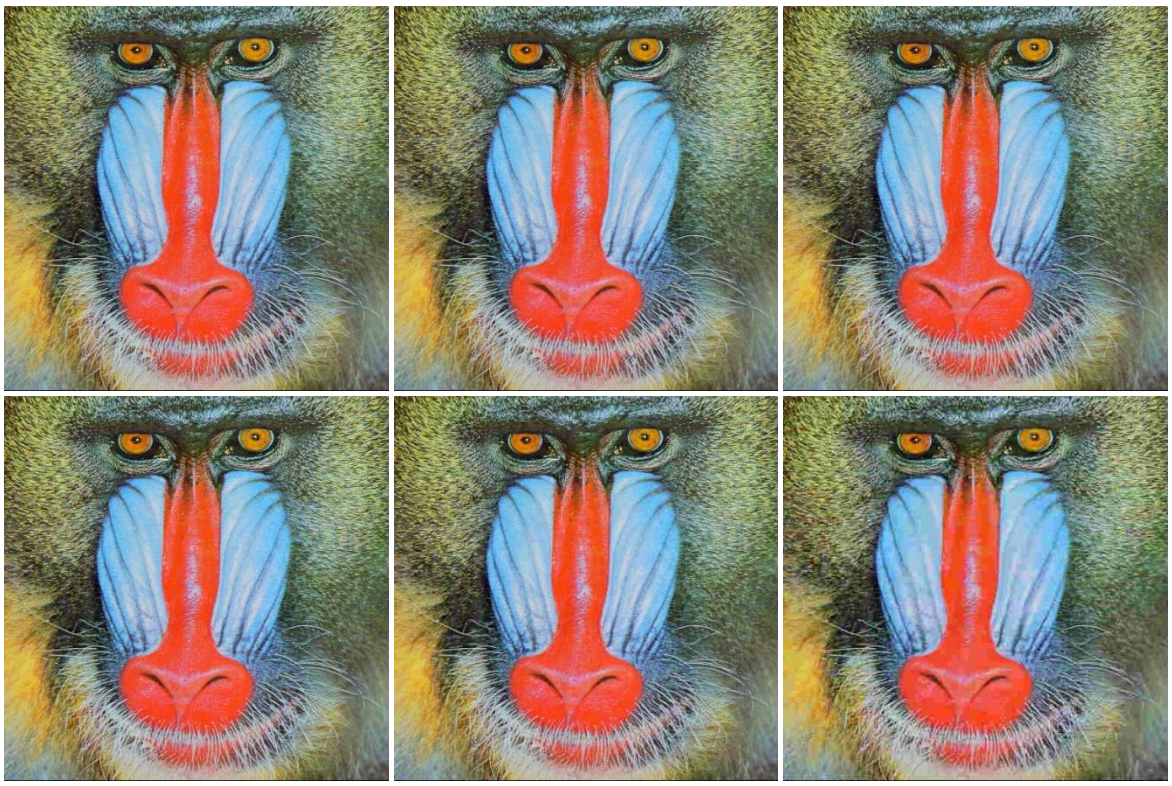
Οι αντίστροφες διαδικασίες είναι προφανείς, οπότε δεν εξηγούνται. Σημειώνεται ότι οι πίνακες που αναφέρονται δίνονται από το πρότυπο, και είναι διαφορετικοί για DC και AC συντελεστές και διαφορετικοί για τις συνιστώσες Y και Cr, Cb. Έτσι, υπάρχουν συνολικά 3 x 2 = 6 πίνακες, οι οποίοι βρίσκονται στο αρχείο huffman_tables.py.

Ενσωμάτωση JPEG

Σε αυτό το στάδιο ζητείται η κατασκευή συναρτήσεων κωδικοποίησης και αποκωδικοποίησης μιας εικόνας σύμφωνα με το πρότυπο JPEG, χρησιμοποιώντας τους προαναφερθέντες μετασχηματισμούς. Η υλοποίηση βρίσκεται στο αρχείο jpeg_encoder.py.

Αποτελέσματα

Ζητείται να παρουσιαστούν τα αποτελέσματα για διάφορες τιμές του qScale.

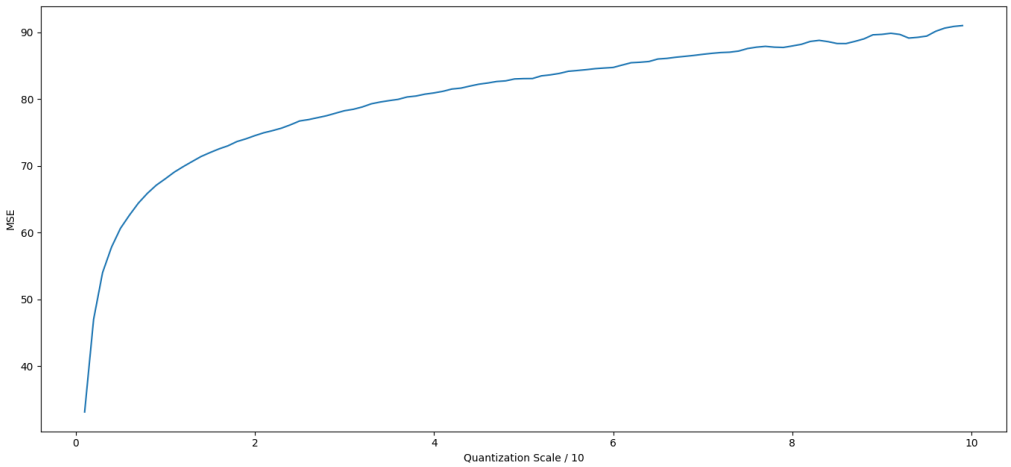


Εικόνα 3. Διάφορες τιμές κλίμακας κβαντισμού για την εικόνα 1

Εικόνα 4. Διάφορες τιμές κλίμακας κβαντισμού για την εικόνα 2

Τα παραπάνω αποτελέσματα φαίνονται αναλυτικότερα στον φάκελο images/qScales.

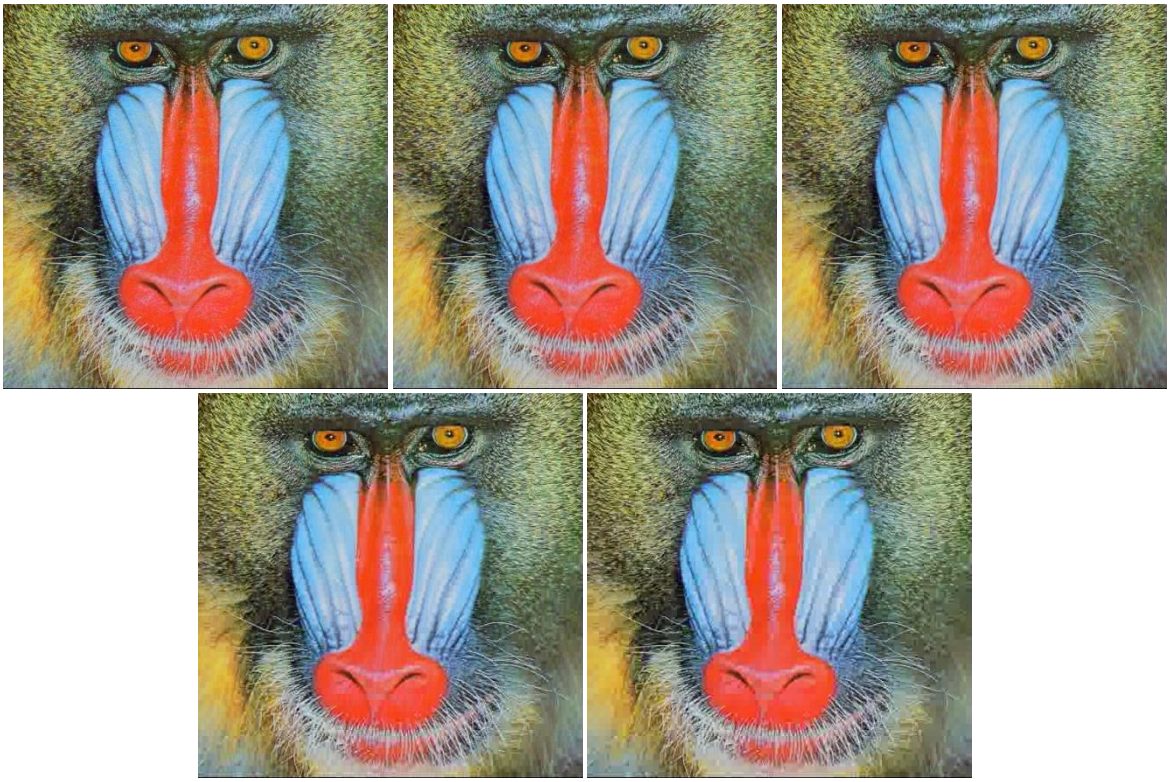
Παρακάτω φαίνεται το μέσο τετραγωνικό σφάλμα για διάφορες τιμές του qScale, από 0.1 έως 10 με βήμα 0.1.



Εικόνα 5. Μέσο τετραγωνικό σφάλμα

Παρατηρείται άνοδος του σφάλματος καθώς αυξάνεται η κλίμακα κβαντισμού. Το φαινόμενο αυτό είναι αναμενόμενο, καθώς με την αύξηση της δεύτερης, μεγαλώνει το περιθώριο στρογγυλοποίησης, άρα και μειώνεται η ακρίβεια της πληροφορίας.

Τέλος, ανακατασκευάστηκαν οι εικόνες, με τροποποιημένες τις κλίμακες κβαντισμού, ώστε να μηδενιστούν οι 20, 40, 50, 60 και 63 πλέον υψίσυχνοι όροι του block, με κλίμακα κβαντισμού 1. Παρακάτω φαίνονται συνοπτικά τα αποτελέσματα, όμως υπάρχουν αναλυτικότερα στον φάκελο images/qTable.



Εικόνα 6. Μηδενισμός συντελεστών DCT για την εικόνα 1

Εικόνα 7. Μηδενισμός συντελεστών DCT για την εικόνα 2

Σημειώνεται ότι όλα τα παραπάνω πειράματα πραγματοποιήθηκαν με υποδειγματοληψία 4:2:2, και για τις δύο εικόνες.

Demo 2

Το τελευταίο ζητούμενο της δεύτερης εργασίας είναι το αρχείο demo2.py, στο οποίο υπολογίζονται κάποιες εντροπίες:

1. στο χρωματικό σύστημα RGB
2. των κβαντισμένων DCT συντελεστών
3. χρησιμοποιώντας τα μήκη διαδρομής

Τα αποτελέσματα φαίνονται παρακάτω, και μπορούν να εμφανιστούν και σε πραγματικό χρόνο στην κονσόλα, με την εκτέλεση του αρχείου.

	Baboon image			**Lena image**		
RGB original	7.7624			7.7502		
RGB reconstructed	7.7739			7.4102		
	Y	**Cr**	**Cb**	**Y**	**Cr**	**Cb**
DCT	2.3185	0.698	0.7581	0.3969	0.1609	0.1406
RLE	5.3567	4.1694	4.3818	2.9985	1.5121	1.5578